

1. What's a microprocessor ?

silicon chip which includes ALU , register circuit and control circuit

هو عبارة عن شريحة أو اقلية تكون من وادي معالجة مركزية تسمى CPU وهي إختصاراً لـ "central processing unit" وهي وادي ربط تربط المعالج مع الوسط الخارجي والمنطقي .

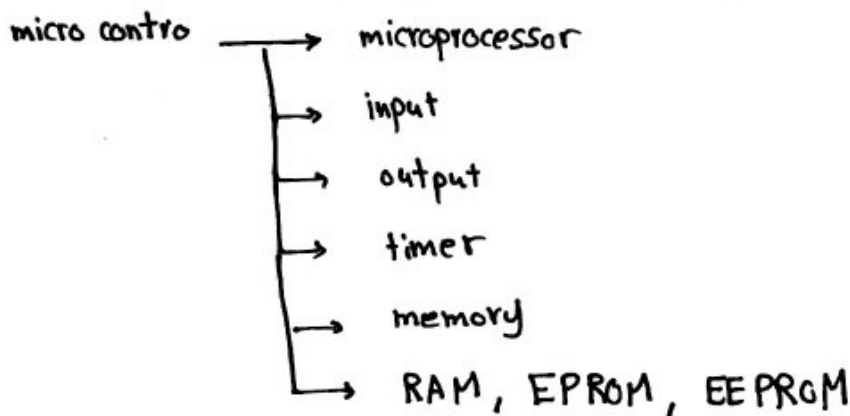
ALU : وحدة معالجة تقوم بإجراء العمليات الحسابية

register : مسجل يخزن البيانات المؤقتة

CU : control unit إشارات التحكم

2. What's a micro controller ?

هو دمج وادي ومعالج ال microprocessor ثم يجمع ملحقات ال processor ووضعها في شريحة واحدة ، تعتبر شريحة مدمجة وليست معلقة .



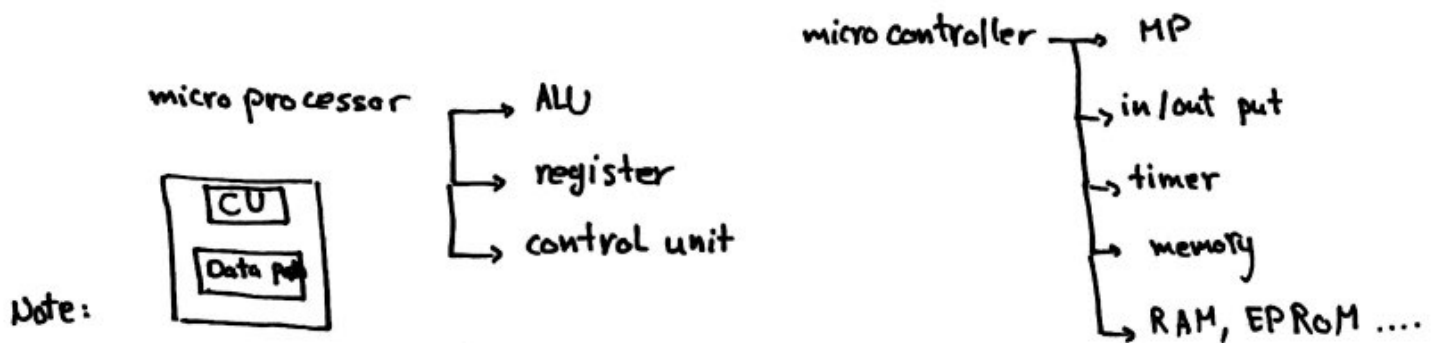
* microprocessor based system :-

Data Bus " هو الذي تبعت منه بيانات "

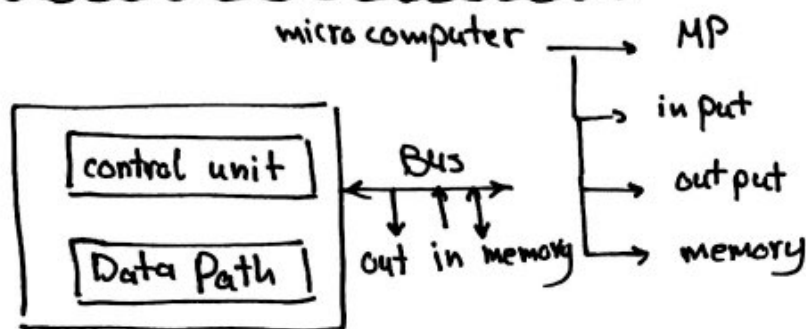
control Bus " تقرأ أنك تقرأ أو تكتب "

Address Bus " هو الذي يكمل العنوان "

3- What is the difference between " microprocessor, microcomputer, micro controller " ?

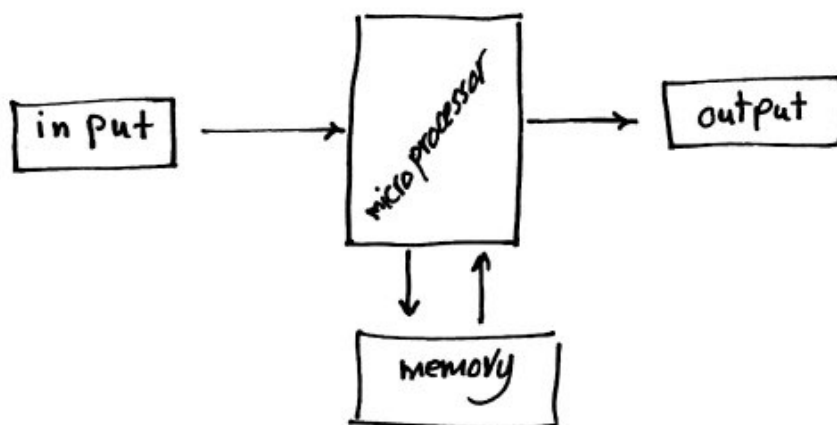


" Data path = ALU and register "



4- Draw the microprocessor based system?

microprocessor based system = microcomputer



2

5. Explain the microprocessor system bus?

A Bus is set of common connections that carry the same type of information .

In every computer there are three types of busses:

Data bus , Address bus , control bus .

Bus: عبارة عن رابط لنقل البيانات

6 - What's meant by :

a) Instruction :-

هي قائمة بجميع الأوامر المتاحة بمختلف أشكالها التي يمكن لمعالج ما أن يقوم بتنفيذها , تتضمن هذه التعليمات " حسابية , منطقية , نقل , تكلم "

b) Machine Language :-

هي اللغة الوسيطة التي يفهمها وينفذها الحاسوب مباشرة , يكون قاموسها على كثير من الاينكازات , عند كتابة برنامج بلغة ++C مثلا ! يتم تحويل هذه الأوامر إلى لغة التجميع الخاصة بالمعالج ومن ثم إلى لغة الآلة حتى يتم تنفيذها !

c) Assembly Language :-

هي مجموعة من اللغات تستخدم في برمجة أجهزة الحاسوب , تقدم هذه اللغة بتحويل الثوابت أو الأكواد اللازمة لبناء برنامج معين من « CPU » من شكله المعتمد على الرموز إلى شكل آخر يسمى كود الآلة

d) Assembler :-

يستخدم المجمع في ترجمة السطور والتعليمات المكتوبة إلى كود الآلة
من شكل مكتوب بلغة التجميع ليحول إلى شكل مكتوب بلغة الآلة

3

e) compilers =

يُحوّل برنامج من شكل إلى آخر ، عند طلب تشغيل الكود يُفسّر الكود التنفيذي ولا يخبر الترجمة من البايثون ، من خصائص الـ compiler انه يستهلك وقت كبير في فترته وتحويل الكود الأصلي إلى الكود الثنائي.

7- What Language a microprocessor understands?
Machine Language

8- ~~What are the width of data bus and address bus of 8086?~~

9- What are the width of data bus "DB" and address bus AB of 8086?

Data bus: 16 bit , Address bus: 20 bit

$$* \quad 2^1 = 2 , \quad 2^2 = 4 , \quad 2^3 = 8 , \quad 2^4 = 16 , \quad 2^5 = 32$$

$$2^6 = 64 , \quad 2^7 = 128 , \quad 2^8 = 256 , \quad 2^9 = 512 , \quad 2^{10} = 1024$$

: CPU time:

$$\text{cpu time} = I_n \times \text{CPI} \times \text{clock cycle}$$

$$\text{clock cycle} = T = \frac{1}{F}$$

4

ex: $F = 1\text{kHz}$, $I_n = 10\text{ms}$, $Phy = 20 I_n$

Find the cpu time:—

$$T = \frac{1}{F} = \frac{1}{1\text{kHz}} = 1\text{ms} \rightarrow T = 1\text{ms}$$

$$I_n = 10\text{ms} \rightarrow CPI = \frac{I_n}{T} = \frac{10(\text{ms})(\text{cycle})}{1\text{ms}}$$

$$CPI = 10\text{ cycle}$$

اللي في القانون $I_n = 20 I_n$

تحمل عدد التعليمات
يعني 20 أمراً أو 30

$$\text{cpu time} = 20 \times 10 \times 1 \times 10^{-3} = 200 \times 10^{-3}$$

$$\text{cpu time} = 200\text{ms}$$

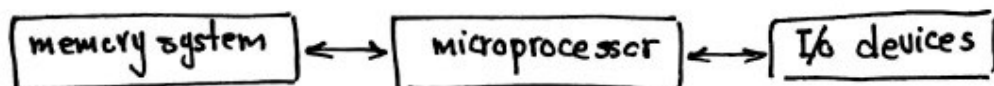
* Data types:—

Like binary , BCD, ASCII , signed , unsigned number.

10. What determines that microprocessor is an 8 bit , 16, 32bit?
is the size of the data path " data path \rightarrow (ALU and reg)

11. List the components of a computer also draw a block diagram of personal computer?

microprocessor , memory system , I/O devices
 \rightarrow input and output



12. Define bit, byte, word, double word, quad word?

Bit: zero or one, false or true.

Byte: is an 8 bit

word: is a 2 byte

double word: is 4 byte

quad word: is 4 words

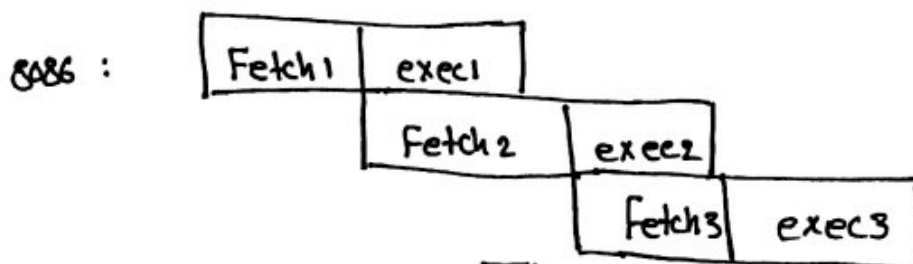
13. What's the difference between 8085 and 8086?

8085	8086
data bus: 8 bit	data bus: 16 bit
address bus: 16 bit	address bus: 20 bit
3 MHz	5 or 10 Hz
6500 transistor	29000 transistor
64 k byte	1 M byte
It does not support pipelining	It support pipelining

14. Explain the concept of pipelining in 8086? Draw a Block diagram?

Pipelining is the simplest form to allow the CPU to Fetch and execute at the same time.

That the fetch and execute times can be different.



15. An assembly language program is translated to machine code by: a) assembler b) compiler c) linker d) interpreter

16. The 8086 used two processing logical units with which were known as: a) segment and offset units

b) Bus Interface unit and Execution unit c) ALU and control unit

d) Bus unit and Execution interface unit

17. The Arithmetic logic unit "ALU" it can do the:

a) Addition b) subtraction c) multiplication d) Division
e) all the above

18. What's the architecture of 8086 includes:

Arithmetic logic unit (ALU), Flags, registers, Instruction queue register, Segment register

-: Bus Interface Unit :-
"BIU"

8086 or 8088 memory interface concept:

- Instruction pointer, Instruction queue register, Segment register

• Instruction pointer, IP,

contains the offset of the next instruction to be executed

عنوان الحالة التالية يبدأ في الـ [IP]

[7]

- Instruction queue register :

- The instruction size vary From 1 to 6 byte .
- The BIU Feeds the instruction stream to the execution unit through a 6 byte instruction unit .
- The BIU store the per fetched instruction in a first in first out register .
- EU simply reads the next instruction byte form the queue register in BIU

بمعنى ان BIU هو الأب الذي يتعامل مع الولد الخارجي ويجب الأوامر والمسلزمات
يوصلها له طريق Queue لا EU الذي تعتبر كأنها الأم
BIU ← تمتلئ Queue بأنفا الأبطال ، BIU يبطل خدمة لحاضار مشغولين
تفقد الأوامر " يعني Queue ممتلئ ان BIU واقف "
EU ← تمتلئ Queue كأنها المطبخ ، لانها لما المطبخ فاضي ال EU متقدمه غير
خدمة وواقفة لان ما عندها ما أدير ! " يعني Queue فاضي EU واقف "

- segment register :-

- * The memory of 8086 is divided in to four segment
Code segment «CS»
Stack segment «SS»
Data segment «DS»
Extra segment «ES»

* segment register → 14 register → 16 bit
✍

8

- Code segment : يخصص هذا المقطع لتخزين عنوان البرنامج
- Data segment : " " " " " البيانات والمعلومات
- Stack segment : يستعمل للحفظ المؤقت ، لمخزن المعلومات الضرورية التي لا يفيش أن تضيع أو تتغير أثناء تنفيذ البرنامج ، آلية العمل تكون وفق قانون «Last in First out» يتم سحب المعلومة من ال Stack من قمة حيث يتم فيه العاجلة لاستعمال مقطع مخبرات بنفس الوقت
- Extra segment : وذلك لكي نستطيع الاستفادة من أكبر مساحة في الذاكرة.

CS : يحدد أول location في الذاكرة
 DS : " " عنوان Data
 SS : " " Location في stack memory

* كل segment حجمة 64k يبدأ من 0000 إلى FFFF
 16 segment = 1Mb

19 - How many address line dose an 8086 have?
 20 Address Line

20 - How many memory addresses this number of address lines allow 8086 to access directly?

$$\frac{N}{2} = \frac{20}{2} \text{ Address, } n = \text{number of address lines}$$

21 - at any given time , the 8086 works with Four segment in this address space , How many byte are contained in each segment?

64 kbyte on each segment .

22. What's meant by the term "machine cycle" ?
عدد المرات التي يتكرر فيها لا code segment في ال memory
تجيب بيانات تتكرر (machine code)

23. Distinguish between (KB, MB, GB, TB) ?

- 1 kilobyte is 2^{10} bytes
- 1 megabyte is 2^{20} bytes
- 1 gigabyte is 2^{30} bytes
- 1 tera byte is 2^{40} bytes

24. What is meant by the statement that 8085 is a 8 bit microprocessor?

8 bit = register كل

* control Lines: -

- MRDC (memory read control)
- MWTC (memory write control)
- IORC (I/O read control)
- IOWC (I/O write control)

* micro computer memory can be logically divided into
three groups:—

1. Process memory «Register»
2. Primary memory «Rom - Ram»
3. Secondary memory («magnetic tapes and disks»)



• Bus interface unit contains:

- segment Register
- Instruction Pointer
- 6 Byte instruction Queue
- Address adder

-: Execution Unit contains:-

General Registers

Base Pointer

Stack Pointer

Index Pointer

ALU

Flag Register

Instruction Decoder

Timing and control Unit

execution unit:

وحدة تنفيذ مسؤولة عن فك شفرة التعليمات وتنفيذها ، تقوم بجلب التعليمات من BIU وتقوم بفك شفرتها والعمل عليها .
تحتوي على عدد أربع مسجلات (Ax, Bx, Cx, Dx) يتم استخدام هذه المسجلات في التعامل مع البيانات داخل المعالج .
يستطيع المعالج ان يتعامل مع بيانات في الذاكرة ، إلا أن التعامل مع المسجلات يكون أسرع بكثير .

25 - Describe in detail the general purpose of data registers?

- Data register = Dx

يقتر كأنه ملحق لـ Ax بمعنى يمكن وضع باقي النواحي وتخزينها به من الممكن تخزين بعض البيانات أو الثوابت

- Base register = Bx

يستخدم المسجل Bx في كونة الذاكرة حيث تتطلب بعض العمليات التعامل مع الذاكرة بجوهر ماكد .

- Count register = Cx

يستخدم كعداد التكرار بعد المرات

- Accumulator register = Ax

يقتر هو المسجل المفضل للاستخدام في العمليات الحسابية والمنطقية و تعمل البيانات والتعامل مع الذاكرة

* stack pointer : مؤشر آخر موقع بيانات

* stack segment register : يحدد أول موقع

* Base pointer : مؤشر يأت على الذاكرة

مختصين بالسلاسل و $DI, SI \Rightarrow$

المصنفات

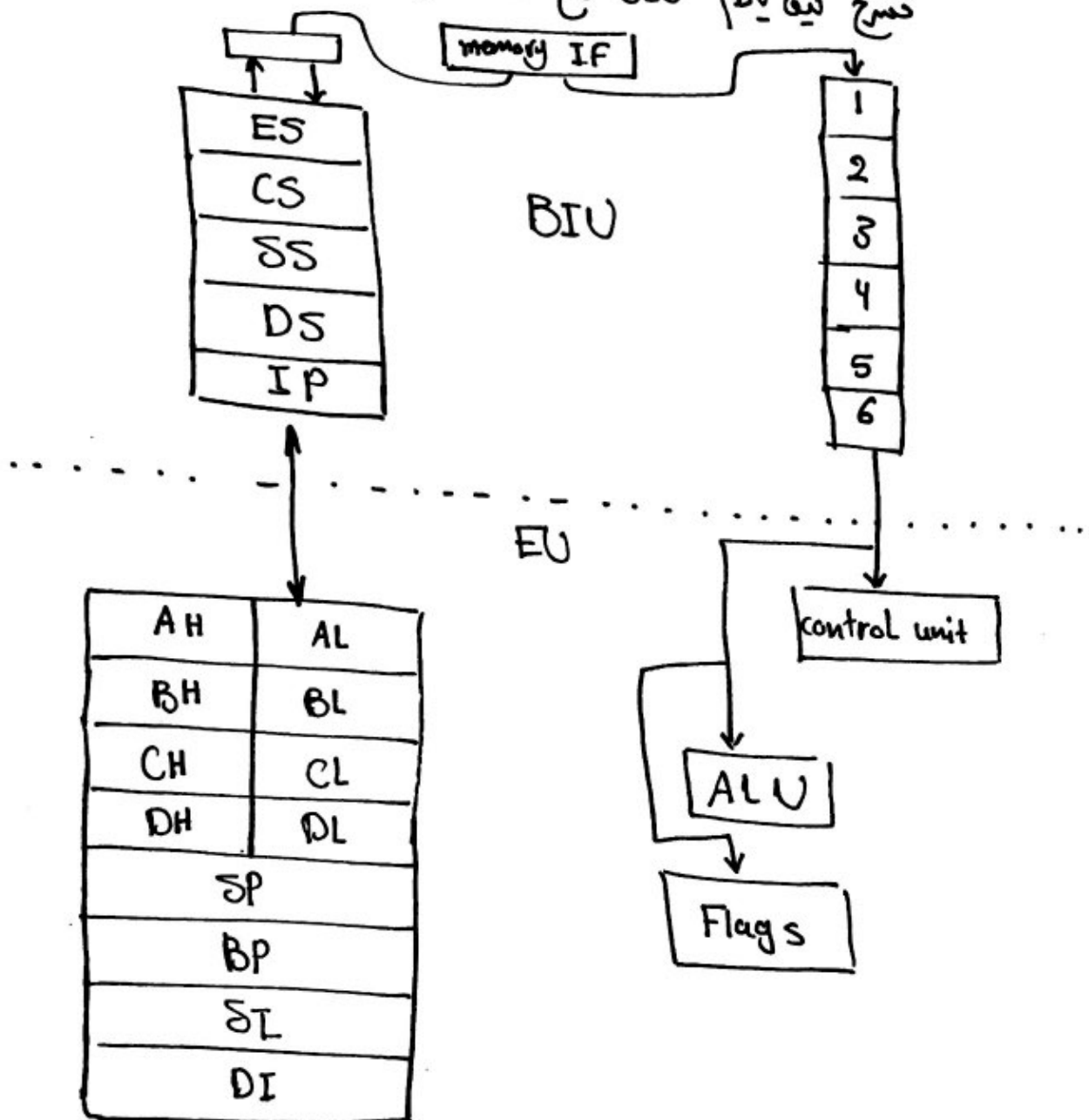
Source index : لتأثير على سلسلة ترتيب القياسات

Destination index : يؤثر على واجهة string

26- Draw and explain the architecture of 8086?

The architecture of 8086 includes:

ALU, Flags, Registers, instruction queue Register, Segment - .
 تشرح كيف يفصل BIU عن EU; في ما شرحته قبل!



13

26 - What's the function of a segment register in 8086?

Each segment contains the starting address of the corresponding memory segment block.

كل segment يحتوي على عنوان Block في ال memory

27 - Explain the concept of segmented memory?

Segmentation:-

64kb memory مقسم الى 16 Blocks وكل Block بحجم 4kb

كل Block يختلف الآخر من حيث الالوان والاستخدام مثلا:

code segment block contain only code

data segment block contain the data of the programs.

27 - List the following 8086 registers:

The Four 16 bit general registers: AX, BX, CX, DX

The two index registers: SI, DI

The three pointer registers: SP, BP, IP

The Four Segment registers: CS, DS, SS, ES

28 - Which segment contains the executable machine code?

Code Segment « CS »

* 29 - List the function of Bus interface unit in 8086?

Sends out address

Fetches instruction From memory

Reads data from ports and memory

Write data to port and memory

30. Since the x86 has an bus of 20-bits, its memory is segmented into 4M segment:

a) True b) False

31. pipelining improves CPU performance due to:
increased clock speed

32. A 32 bit address register can access up to —, of memory: 2kB, 2GB, 4GB, 6GB

33. What are the names of the four segment registers?

a) Data, Index, Code, Stack

b) Stack, Extra, Data, Code

c) Stack, Data, Base, Counter

d) Stack, Index, Extra, Code

34. What is the largest signed integer that may be stored in 32 bits?

a) $2^{32} - 1$

b) 2^{32}

c) $2^{31} - 1$

d) 2^{31}

unsigned $\rightarrow 2^{32}$

35. List all of the 8086 eight bit registers:

AL, AH, BL, BH, CL, CH, DL, DH

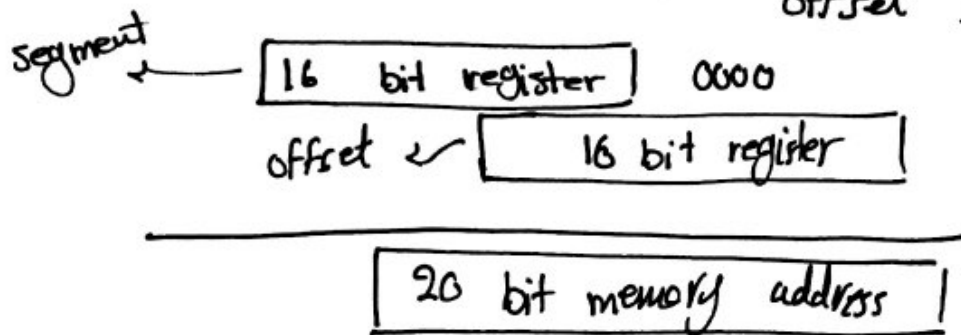
15

Segment	offset	
CS	IP	→ instruction
SS	SP or BP	→ stack address
DS	BX or DI or SI	→ Data address
ES	DI	→ string destination address

36 - How does 8086 convert a logical address to physical address? Explain using diagram?

نظرب ال segment إلى 10 bits حيزا مع ال offset

بدين نزيه ال offset



Ex :

CS = 4370 , IP = 1523

physical address = 43700 + 1523 = 45223

//

Logical address → Segment : offset

physical address → (Segment) * (10) + offset

Offset address → offset

Lower range → (Segment) * (10)

Upper range → (Segment) * (10) + FFFF

16

—: physical add... as the address

37. IF CS = 24F6H and IP = 634AH show:

The Logical address, offset address, physical address
lower range, upper range?

Logical address = Segment : offset = CS : IP = 24F6 : 634A

offset address = IP = 634A

physical address = (24F6) * (16) + 634A = 2B2AA

lower range = (Segment) * (16) = 24F60

upper range = 24F60

38. Assume that the DS register is 578C, To access
a given byte of data at physical memory location 67F66,
does the data segment cover the range where the
data is located?

No, since the range is 578C0 to 678BF
Location 67F66 is not included in these range.

—: Flags:—

- يوجد نوعان من Flags نوع يصف حالة الناتج ونوع ليع كالاتي
بشكل او يصف التحكم أثناء البرنامج

Condition Flags
→ carry flag, zero flag
→ sign flag, over flow
→ parity flag, Auxiliary carry flag

Control Flags → Trap flag
→ interrupt
→ Direction

1 - Carry Flag «CF»

لو كان الجمع كذلك carry بعد العملية الحسابية معناها $CF=1$

2 - Zero Flag «ZF»

لو كان الناتج كله كان أصفار معناها $ZF=1$ لو كان في حق واحد واحد معناها $ZF=0$

3 - Auxiliary Flag «AF»

لو جمع كذلك carry ورفعت من Byte 1 إلى Byte 2 أو من Byte 3 إلى Byte 4 أو من Byte 2 إلى Byte 3 معناها $AF=1$ حتمهم أكثر بالأمثلة!

4 - Parity Flag «PF»

لشوف عدد الواحدات في أول 8 bits لو كان عددهم فردي معناها $PF=0$ لو كان زوجي $PF=1$

5 - Sign Flag «SF»

نشوف الناتج لو كان آخر bit ع اليسار اللي MSB واحد معناها العدد مثل بإشارة ويكونه $SF=1$ ولو كان آخر bit ع اليسار 0 معناها $SF=0$

6 - Overflow «OF»

لو كان في overflow معناها $OF=1$

Control Flags: -

Trap Flag (TF)

it is used for single step control

يعني لما تكتب برنامج وتبدأ في كل خطوة تتشوف شئ صار يعني تتحرك بالسطر الـ $TF=1$

39. If the following addition is carried out:

$$\begin{array}{r}
 \begin{array}{cccc}
 \text{0} & \text{0} & & \text{0} \\
 0010 & 0011 & 0100 & 0101 \\
 0011 & 0010 & 0001 & 1001 \\
 \hline
 0101 & 0101 & 0101 & 1110
 \end{array}
 \end{array}$$

SF = zero → لأنه آخر البت عاكس
 ZF = zero → لأنه الناتج مساو للصفر
 PF = 0 ← عد الوحدات في أول 8 bits خمسة منها ما
 AF = 0 ← لما انتقلت من Byte ل Byte مرزقتس مكان
 OF = 0 ← لأنه ما فيش over flow

$$\begin{array}{r}
 \begin{array}{cccc}
 \text{0} & \text{0} & \text{0000} & \\
 0101 & 0100 & 0011 & 1001 \\
 0100 & 0101 & 0110 & 1010 \\
 \hline
 1001 & 1001 & 1010 & 0011
 \end{array}
 \end{array}$$

ZF = zero PF = one OF = one
 SF = one AF = one CF = zero

MOV BH, 38H

ADD BH, 2FH

$$\begin{array}{r}
 \begin{array}{cc}
 38H \rightarrow & \begin{array}{cc} \text{0} & \text{0} \end{array} \\
 2FH \rightarrow & \begin{array}{cc} \text{0} & \text{0} \end{array} \\
 67 = & \begin{array}{cc} \text{0} & \text{0} \end{array}
 \end{array}
 \begin{array}{r}
 \begin{array}{cc}
 0011 & 1000 \\
 0010 & 1111 \\
 \hline
 0110 & 0111
 \end{array}
 \end{array}$$

CF = zero PF = zero
 ZF = zero AF = one
 SF = zero

19

40. Flag register of 8086 («status Flags») or «condition Flags»
CF, SF, OF, ZF, AF, PF
41. Flag register of 8086 «control Flags»
TF, DF, IF
42. by default CS is associated with:
a) SS b) BP c) IP d) CX
43. IF CS = 7FA2H, SS = 0801H, SI = 0100H, IP = 438EH
the address of the next instruction is:
a) 83DAEH b) 438EH c) 83DA0H d) None of the above
44. Which of the following is the pair of register used to access memory in string instruction?
a) DI and BP b) SI and BP
c) DI and SI d) DS and SI
45. IF ES = D321H, then the range of physical address of the extra segment is:
a) 00000 - 0D321H
b) D3210H - D321FH
c) D3210H - E320FH
d) 0D321H - 1D320H

20

CS: 8FA1F
SI: 1800F
DI: 0110

46. IF the segment For an 8086 program start at address 70400H, What number will be in the CS register?

Segment program start at address = (Segment) * 16
Segment = CS = 7040H

47. What's the physical address? offset address?
Logical address?

- physical address:

is a 20 bit address that actually put on the address bus, Has range 00000H - FFFFFH

- offset address:

is location within 64kb segment range, Has a range of 0000H - FFFFH

- Logical address :

consists of segment address and offset address

48. If the content of the register SS = 3500H and the content of the register SP = FFFFH then calculate:-

- physical address: $35000 + FFFF = 44FFFH$

- Logical address: (3500:FFFF)

- lower range of the stack: (35000H)

- upper range of the stack: $35000 + FFFF = 44FFFH$

49- Describe the function of the 8086 Queue?

And does the queue speed up processing?

BIU يبعث التعليمات إلى EU من طريق Queue ،
EU فترا بوضع ثمن ومعلمها وتبدأ تنفذ فيه وفي نفس الوقت
تبدأ ال Queue مونت التعليمات التالية من BIU ، يرغب بكل
بساطة ال Queue هي حلقة الوصل بين BIU و EU
أكبر زادت من سرعة ال processor لأنه تغطي في الكمية الحالية
وتكون مونت الكمية المتأخرات .

50- one of the following is not a valid segment address:

- a) 00000 b) E0840 c) 8CE90 d) 8CE91

—: Addressing modes: —

- The addressing mode available in Intel 8086 are:

- 1- Register Addressing
- 2- Immediate Addressing
- 3- Implied Addressing
- 4- Direct Addressing
- 5- Register Indirect Addressing
- 6- Based Relative Addressing
- 7- Indexed Relative Addressing
- 8- Based Indexed Relative Addressing

51. What an instruction consists of?

- An instruction consists of an operation code "op code" and the address of the data "operand".

instruction \Rightarrow operation , operand
العملية \Rightarrow العمل

ex: -

MOV AX, 1000H

MOV is the op code

AX Register is an operand

* AX Register is an operand
Op code \Rightarrow 1 Byte, Operand \Rightarrow 1 or 2 Byte

52. What are different instruction formats?

a)

operation code

Implied

b)

operation code	operand
----------------	---------

Immediate operand

c)

operation code	Direct address
----------------	----------------

Direct Addressing

d)

operation code	Indirect
----------------	----------

Indirect Addressing

53. What is opcode Fetch cycle?

Each instruction starts with opcode fetch machine cycle

opcode fetch is a machine cycle executed to Fetch the opcode of an instruction stored in memory.

هذه عبارة عن machine cycle
عند سحب opcode المخزن في ال memory تكون cycle

- Source is to the right and Destination the left.

MOV AX, BX
 Destination \downarrow \uparrow Source

- Register addressing mode:

Assembly Language	Size	operation
MOV AL, BL	8 bits	copies BL into AL
MOV CH, CL	8 bits	copies CL in CH
MOV AX, CX	16 bits	copies CX into AX

ex: - • MOV CX, SI

لننقل من قيمة ال CX كانت 1234H
 و SI 5678H سيكون الناتج في السجل التالي

before after
 CX 1234H 5678H

Before
 • MOV DL, AH DL 89H BCH
 after
BCH BCH

هنا ننقل قيمة ال high لاننا نعامل مع AH وليس AX

- Immediate addressing mode:—

Examples

- MOV BL, 44 → 8 bits copies a 44 decimal (2CH) into BL
« 44 دېكېمال ۲۲H »
« Decimal 44 »
- MOV AL, 'A' → 8 bits copies an ASCII A into AL
- Value Equ 35H
MOV BH, value

- Implied addressing mode :

مەراتبىيلىك ۋە ۋەزىيەت!

- Instruction using this mode have no operands ✓

Examples:

- CLC → which clears carry flag to zero
- STD → which set direction flag to one

- Direct addressing mode:

- MOV AL, [2C00h]

بۇ ۋەزىيەتتە ۋەزىيەت ۋەزىيەت

8-bits copies the byte contents of data segment memory location 2C00h into AL

- MOV ES: [2000H], AL

8-bits copies AL into extra data segment memory location 2000H

26

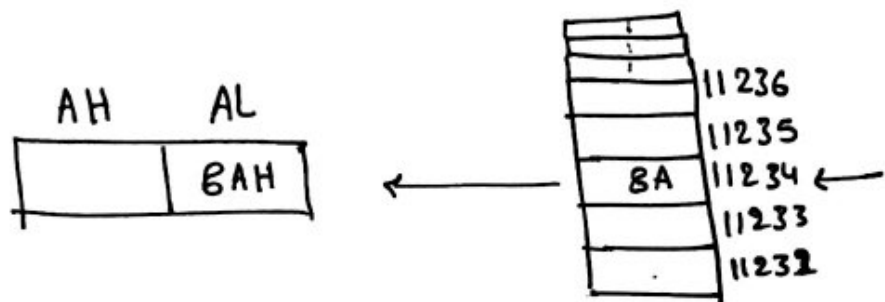
• MOV AL, [1234H] DS=1000H

مفاهما برا لا memory اللى عنوانه 1234H وجيب البيانات
اللى فيه وحطهم في ال AL
لكم حط في بالك حاسب ال physical address بيستعرف
المكان اللى ياتر الكنوان

DS=1000H

$$\text{Physical address} = (\text{segment}) * (16) + \text{offset}$$

$$" " = 10000 + 1234 = 11234H$$

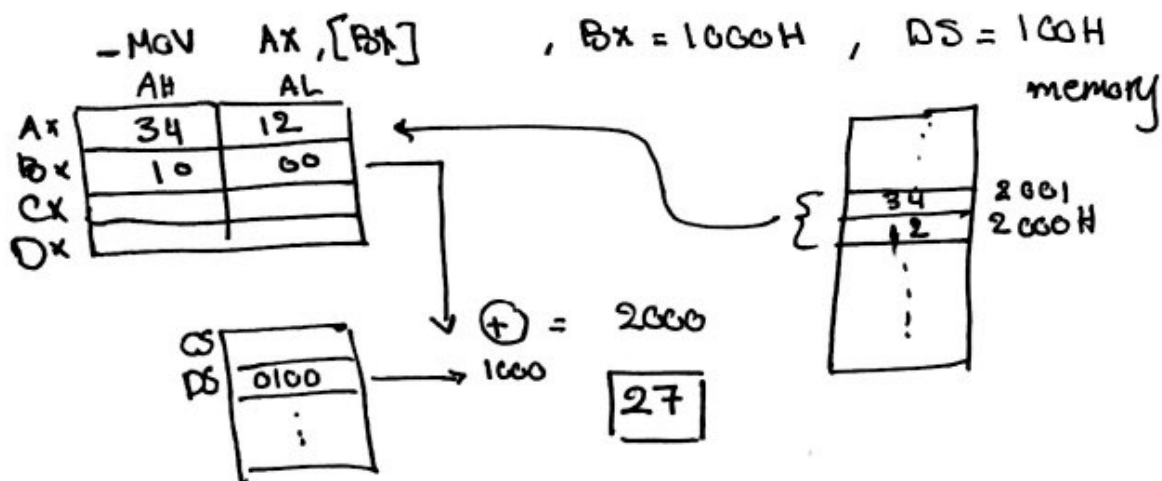


• Register indirect addressing mode: -

- MOV AL, [BX] → moves into AL the contents of the memory location pointed to by DS:BX

مفاهما برا لا register BX وشوف شن فيه وبكدين امشي حطه في AL

- MOV CL, [SI] → moves contents of DS:SI into cl



- Based Addressing with displacement:
- Based relative addressing mode:

فيه هذا النوع خذنا يا إما بـ BX او BP لا يتك

Based

MOV CX, [BX] + 10

or

MOV CX, 10 [BX]

الزوز نفس المعنى

move DS:BX+10 and DS:BX+11

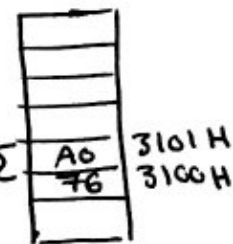
MOV AL, [BX + 1000H] when BX = 0100H

DS = 0200H

AX	A0	76
BX	01	00

1000H → ⊕ → 03100H

DS * 10H = 2000H



- Indexed relative addressing mode:

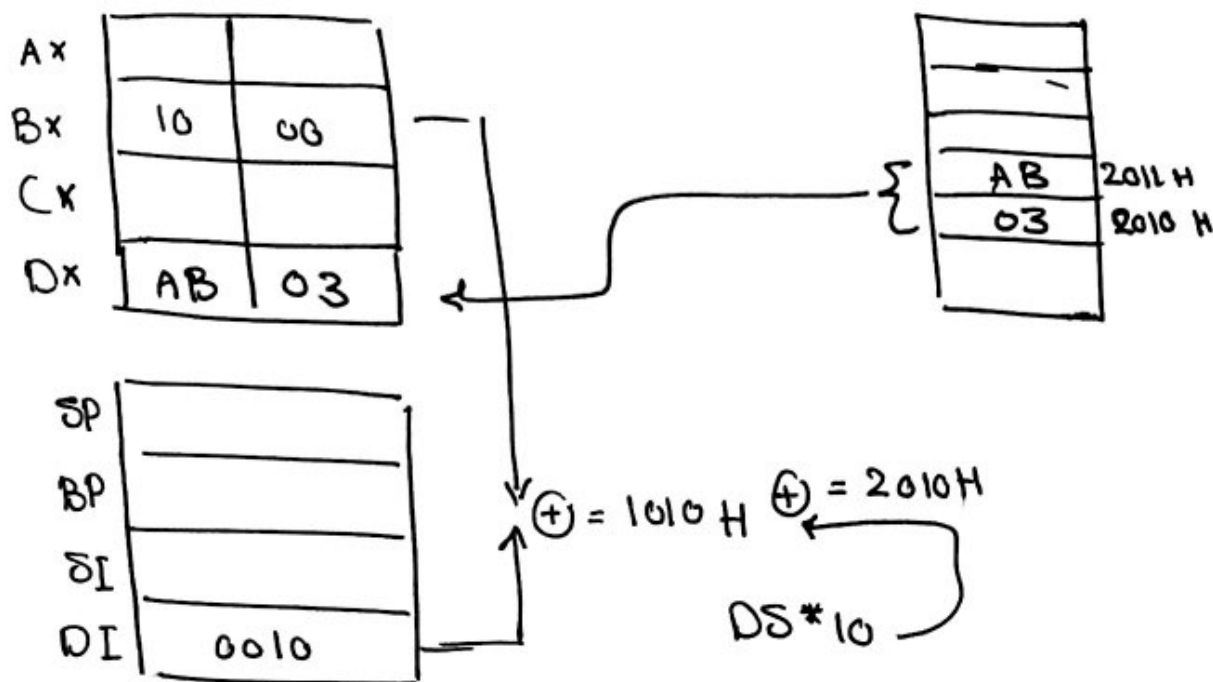
Based Indexed

Indexed Addressing with displacement
فيه الـ

- Based Base relative Indexed addressing mode:
example:—

MOV DX, [BX + DI]

when DS = 0100H, BX = 1000H, DI = 0010H



with displacement 10 11 11 11

- 54 Which of the following is an invalid instruction?
a) MOV AX, CS b) MOV AX, FFFH
c) MOV AX, [BP] d) MOV DS, CS

55. From quickest to slowest instruction execution time, order these three addressing modes:

- a) Immediate, indirect, direct
- b) Immediate, direct, indirect
- c) Direct, indirect, immediate
- d) indirect, immediate, direct

29

56. In general, the source operand of an instruction can be:

- a) memory location b) register c) immediate data
- d) all the above

57. The stack pointer register contains:

- a) address of the stack segment
- b) pointer address of the stack segment
- c) offset of address of stack segment
- d) data present in the stack segment

57. Describe the operation that 8086 will perform when it executes each of the instructions:-

- 1. MOV BX, 03FFh → BX ← 03FFh, Immediate → 16 bits
- 2. MOV AL, 0DBh → AL ← 0DBh, 8 bits
- 3. MOV DH, CL → DH ← CL, Reg to Reg → 8 bits

58. Write the 8086 assembly language statement which will perform the following operation:

- 1. Load the number 7986H into the BP register: MOV BP, 7986H
- 2. Copy BP register contents to the SP register: MOV SP, BP
- 3. Load the number F3H into AL register: MOV AL, F3H

59. Let X1 be an array of words, one of the following is a correct code to set the fifth element in X1 to FF:

- a) mov X1+5, FFh c) mov X1+4, FFh
- b) mov X1+10, FFh [30] d) mov X1+8, FFh

60. The instruction `MOV AX, X1[DI]` is an example of:

- a) indexed addressing
- b) indirect addressing
- c) direct addressing
- d) based addressing

61. Which flags are NOT used for mathematical operations?

- a) carry, Interrupt, Trap
- b) Direction, Interrupt, Trap
- c) Direction, overflow, Trap
- d) Direction, Interrupt, Sign

62. Which of the following defines a constant Max?

- a) `Max db 80`
- b) `mov Max, 80`
- c) `Max equ 80`
- d) `Max dw 80`

63. Although 8086 is a 16 bit MP, it deals with 8 bit memory. Why?

- List the 16 bit registers used for register addressing?
`BX, SI, DI, BP`

Q4. What's meant by addressing mode?

An instruction consists of an opcode and an operand.
 لو بتي تتلقى operand من غير و غير !

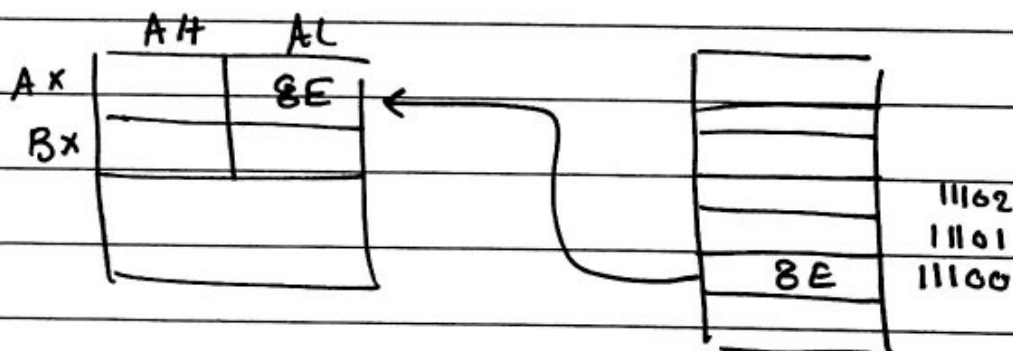
Q5. Suppose that $DS = 1000H$, $SS = 2000H$, $BP = 1000H$ and $DI = 0100H$, Determine address accessed by each of the following: -

`MOV AL, [BP + DI]`

$$BP + DI = 1000H + 0100H = 1100H$$

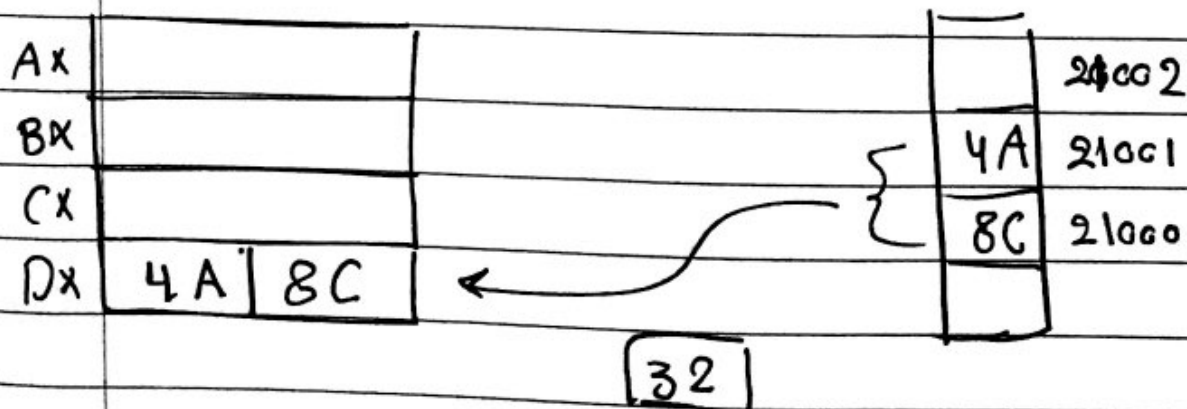
$$DS * 10 = 10000$$

$$(DS * 10) + BP + DI = 11100H$$



`MOV DX, [BP]`

$$BP = 1000H, SS * 10 = 20000 \rightarrow \text{physical} = 21000H$$



—: MOV Instruction:—

mov ينقل بيانات حجمها byte او word من source الى destination

اللي من مصدر الى
memory to memory
Imm to Seg
Seg to Seg

الحالات اللي ممكن نصلي لها بي تكون instruction

MOV AX, BL اول حاجة انه ال size يكون مختلف مثلاً

نلاحظ انه حجم ال des لا يساوي حجم ال src

BL \Rightarrow src

AX \Rightarrow des

MOV SP, 68754h

SP \rightarrow "طوله" حجم 16 bit

68754h \rightarrow 20 bit

وهذا بالنسبة ل size

منسجل نقدر ننقل من memory الى memory

MOV [1234h], [6789h]

منسجل نقدر ننقل من Reg او Seg الى Imm يعني:-

MOV 1235h, Dx

علاش!

لانك انا بي نأخذ مكتوبات Dx الي هوا ال src ونكتبهم في des! اذا كان des /هم كيف بنقل /نصب فيه بيانات
لو كان 1234h بين قولين يعتبر عنوان وتعتبر مح

11

• مهمة في هندسة نقل بيانات لل segment هي طول حتى لو جربت بها بالبرنامج
حيث قولك "Error" مثلاً

خط - MOV DS, 1234h

لو بنى نكولها في instruction مخرج ، او مثلاً بنى نكول بيانات في
segment ، سواء كان «CS, DS, ES, SS» لازم نخرج بيانات في
"register" زي الـ AX مثلاً بدين نكتب في الـ
MOV DS, 1234h هاري instruction الخط هاري
الـ ins مخرج تكون على النحو التالي:-

MOV AX, 1234h

MOV DS, AX ✓

• هندسة نقل مأكولات Seg إلى Seg زي

MOV ES, DS
MOV CS, SS
كم مستحيل

* هناك خطأ بالنسبة لـ size / د بالك تقول أنه MOV AX, 23h خط
لأنه الـ processor يحسب الرقم 23h على أساس 0023h يعني
الـ ins هاري مخرج !

• حتى IP ندم خدمت segment يعني كان بنى نكول فيه حاجة لازم
ننقلها في Reg قبل .

MOV it does not modify flags
في يعني الـ In متأثر على flags

[2]

- مرات يدطيك instruction ويقولك كم عدد operand « المحتامل »

- MOV Ax, Bx one operand
Ax and Bx
- ADD Bx, CX two operand
Bx and CX
- Push SI or POP Ax
one operand
- DAA operand
وهنا

- مرات يدطيك مجموعة instructions ويقولك ما أسرع وأسهل فيهم أو مرات يقولك رتبهم الأسرع ثم الأسرع :-

مثلًا كانت مجموعة instructions كالتالي

- 1) MOV CL, [DI] → Indirect
- 2) MOV CL, [number] → direct
- 3) MOV BL, [BP + DI] → Base index
- 4) MOV CX, [BP + DI + number] → Base index with displacement
- 5) MOV AX, 34h → Imm

أسرع واحد الخامس لأنه لا يحتاج إلى أي شيء في الـ Ax
غير ما يعني لي مكان ، الذي يريده رقم (2) ، الذي هو direct حيث العنوان الذي داخل العتس ويرى في CL ، الذي يريده indirect حيث العنوان الذي يأخذ DI ويرى ما يجب أن physical يريده يأخذ في CL ، الذي يريده الـ Base index

والذي يريده الـ Base index with displacement

الترتيب الذي يليه : Imm, direct, indirect, Base index, Base index with displacement

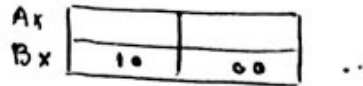
3

- ملاحظة -

عند نقل مكوّنات عنوان إلى Reg يمكننا! كتابة offset ثم الرقم
وسيصبح كل الشئ التالي: -

MOV BX, offset [1000h]

هاري معنا ان قيم BX حتما 1000



لوقالك MOV BX, [1000h] هاري بروحها مليعاش علاقة!

MOV BX, offset [1000h] ≠

MOV BX, [1000h]

MOV BX, offset [1000h] =
MOV BX, offset 1000h

نك :-

رغم في ال source لمأخذ ال offset والرقم لسواد

حتمية بين قوسين اولك حياخذو ويسلف في Reg

- What is the effect of executing the instruction:

MOV CX, [source - MEM]

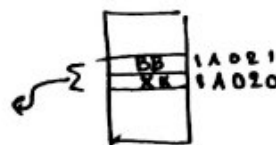
Where source - MEM equal to 20h is a memory location
with DS = 1A00h

رغم ال source - mem تأخذ ال location ال 20h و احنا قلنا ان كل

Location في ال memory طوله 8 bit فقط، واني حذو ال CX طوله 16 bit
مكتوب حاكب ال physical و مرة نأخذ ال CL و بعدين نزيد واحد ونسلف في

CH, و في هكي: $DS * (10) = 1A000 + 0020h = 1A020$

CL = 1A020h, CH = 1A021



MOV DX, CS -> مبدع هاري

لوانت MOV CS, DX خط

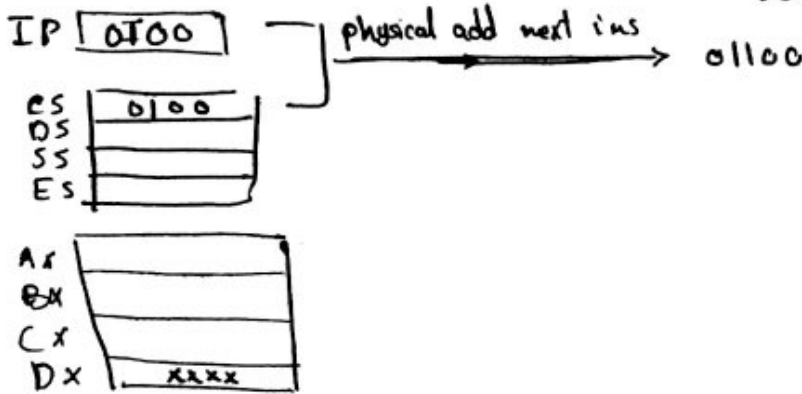
ملا حقة مهمة جداً :-

أني قلت كان بي الخط في segment لازم نخرج الى جة هاري في Reg
بدون نصب Reg في ال seg لك ال segment « CS » مينطبقش
عليها الكلام هذا لاننا نحتاج حاجة اسمها « JMP or CALL » الى جات
هاري من ال مانهيفيش ، لك لازم تعرف أيت instruction كان ال
des فيها CS منها ال instruction هاري XXXXXXXXXX « حتى IP نفس الشيء »

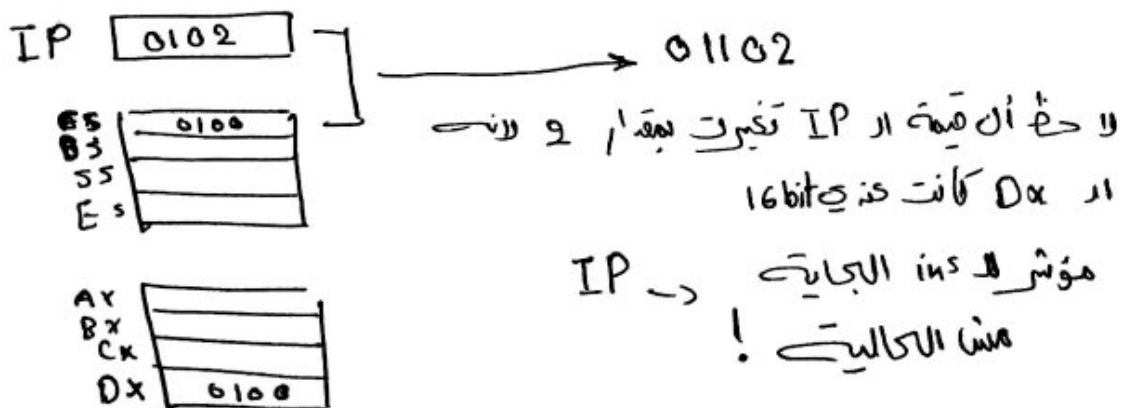
نرجع حالياً للمثال الي هو MOV DS, CS

لم نشن مدناها ال ins هاري؟

Before :-



After:-



5

Byte PTR = For byte

لما بي نخدم بطول one Byte نستخدم In هادي ، و نقدر نستخدم حاجه ممانعة ليها الي هيا « b » يعني

MOV AL, byte PTR [DI] =

MOV AL, b. [DI]

كذلك الحال في ال word

word PTR = w.

MOV AX, word PTR [DI] =

MOV AX, w. [DI]

- Use the mov instruction to copy the contents of memory offset 0300h to memory offset 0500h ?

امني قالك بيك تنسخ مكنويات العنوان 0300h الي 0500h رد بالك
تقول [0500h] MOV [0500h] قلط ! لانه زي ما قلنا النقل من mem الي
mem قلط ، ثبت بنير ؟

MOV AL, [0300h]

MOV [0500h], AL

نرفع المكنويات في Reg بدين نرفع في ال mem

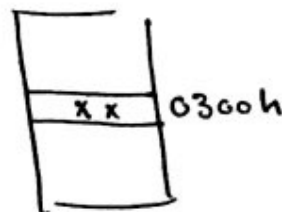
- ايضاً رد بالك تقول أنت كيف استخدمت AL وكذا الي بين

القولين اقم من اربع خانات !

الي بين القولين قلنا ركب عنوان ياشر على Location في ال mem فلوله

8 bit يعني :- mem

يعني صح لانه عنوان مش اقم



6

امني نفس العناين السابق لكم حنينم Ax كيف ا؟

- Use the move instruction to copy the contents of memory offset 0300H and 0301H to memory offset 0500H and 0501H ?

لا location في ال memory يكون 8 bit امني قالك انقل 2 Loc الى
2 Loc , منها حنينم Reg بوله 16 bit لحايني نغز فيه !

MOV AX, [0300H]

MOV [0500H], AX

- write a program to store the data 15H, 75H into memory locations 0055H, 01FCH Assume DS = 7000H?

امني , كز قالك memory location offset ما قالك memory location

منها 0055H من حطها بين قوسين , ال بيبي انك تكون ال 15H في
الحكان ال بيبي (ال physical) ال physical حبيب من DS مع رقم
address

ال location ال قالك ال حنينم ال هو 0055H ← 15H

75H ← 01FCH

كيف ؟ لازم نط في DS قيمة 7000H لكم منقش

ع طوه 15 ما كنا لازم نغز في Reg قبل !

MOV AX, 7000H

MOV DS, AX

MOV BX, 0055H

MOV [BX], 15H

MOV BX, 01FCH

MOV [BX], 75H

7

امني حبيب DS في BX
والسفر ال بيبي حبيب بين قوسين
بيبي لحايني ال physical ال BX متيلو
هني :-

$$DS * (10) = 7000H + 0055H =$$

$$physical = 70055H$$

منها امني ال location رقم 70055H و
في 15H

write a program to exchange contents of memory location offset 3F0ch with that of memory location offset FFC1H ?

بالقالب التالي : memory location offset 3F0ch مع الـ 3F0ch في الـ 3F0ch

هذا البرنامج يهدف إلى تبادل المحتويات بين الـ 3F0ch و الـ FFC1H :

MOV AL, [3F0ch] ; جلب المحتويات من الـ 3F0ch إلى الـ AL

MOV BL, [FFC1H] ; جلب المحتويات من الـ FFC1H إلى الـ BL

MOV [3F0ch], BL ; تخزين المحتويات من الـ BL إلى الـ 3F0ch

MOV [FFC1H], AL ; تخزين المحتويات من الـ AL إلى الـ FFC1H

Xchg : تبادل المحتويات بين الـ AL و الـ AH

سؤال : لو قلنا بدل محتويات الـ AH بـ AL و الـ AL بـ AH ؟

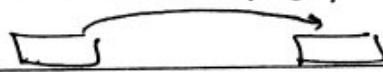
MOV CL, AL

MOV CH, AH

MOV AL, CH

MOV AH, CL

هنا نستخدم الـ CL و الـ CH لتبادل المحتويات بين الـ AL و الـ AH



هذا البرنامج يهدف إلى تبادل المحتويات بين الـ AL و الـ AH باستخدام الـ CL و الـ CH

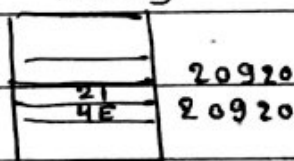
MOV AX, 214EH

MOV [SI], AX

Assume DS = 2042H , SI = 500H

physical address = 20420 + 500H = 20920H

memory



8

—: MOV instruction

- MOV instruction transfers immediate 8 bit data to specified memory location, Assume DS = 4000h ?

memory location 31, 8bit no 50h

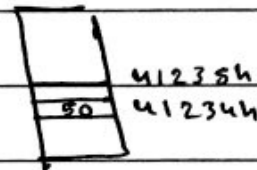
MOV AX, 4000h

MOV DS, AX

MOV [1234h], 50h

memory location ← 50h

physical address = 4000h + 1234h = 41234h



- MOV instruction transfers 8 bit or 16 bit between a register specified memory location :-

Reg 31 mem no 50h
or
mem 31 Reg

mem 31 Reg no 50h

MOV AX, 2000h

MOV DS, AX

MOV SI, 1256h

MOV BL, 55H

MOV [SI], BL

Reg to mem

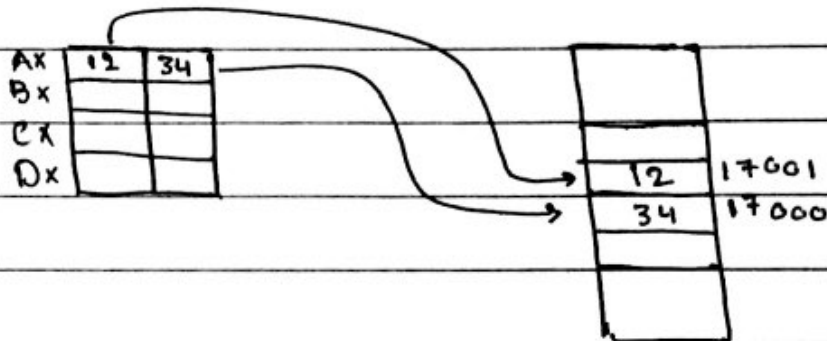
9

• What's the result of `MOV [7000h], AX`?

Assume that the DS register contains 1000H, and AX contains 1234H?

(7000h) : 12 34 AX

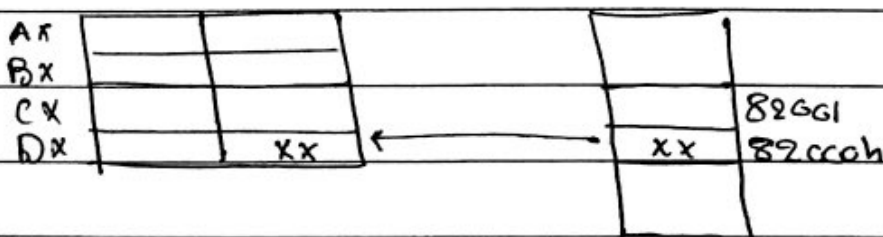
$$\text{physical address} = 10000 + 7000 = 17000h$$



• What's the effective address generated by the instruction `MOV DL, [SI]`, If register SI contains 2000H and the DS register contains 8000H?

(2000h) : xx

$$\text{physical address} = 80000 + 2000 = 82000h$$



—: XCHG :—

• exchange Instruction :—

في instruction لنقل شيء من مكان إلى مكان آخر

Destination , Source

(Ax) ← Accumulator , Reg (16bit)

memory , Register

Register , Register

Register , memory

الأنشغال التي يمكن أن تكون بها xchg

* لا نقول XCHG AX, BX

لأنه في BX قيمة AX وفي AX قيمة BX

XCHG AL, BH

*

في BH قيمة AL وفي AL قيمة BH

* نقل/تبادل بين قيم ال high وال low التي في نفس Reg

رغم ذلك لو كان AX = 1234h نقل/تبادل AX = 3412h

من طريق :— MOV AX, 1234h

XCHG AL, AH ✓

XCHG [SI], DX

*

في هذا مكان محتوي ال physical العنوان [SI]

address

ونستوف من فيه من قيمة وتب لها مكان ال DX

و قيمة DX في المكان الذي يأتي عليه ال physical

← يعني

Assume: DS = 1000, SI = 0500h, DX = 0200h

XCHG [SI], DX

physical address = 10000 + 0500 = 10500h

Before

AX		
BX		
CX		
DX	02	00

BB	10501
XX	10500

after

AX		
BX		
CX		
DX	BB	XX

02	10501h
00	10500h

ex

1. MOV BX, 01FCH

Assume: D = 2000h

2. MOV [BX], 50H

3. MOV AL, 90H

4. XCHG AL, [BX]

في الخطوة الأولى قسّم 01FCH في BX ، في الخطوة

التالية قسّم العنوان الذي يشار له مكتوبات الـ BX في 50H يعني كيتنا

BX = 01FCH \Rightarrow physical address = 20000 + 01FCH

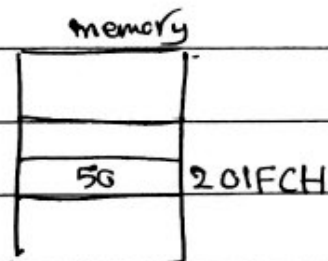
" " = 201FCH

في الخطوة الثالثة قسّم الرقم 90H في

AL ، في الخطوة الرابعة

التيّم الذي في AL و [BX] يعني

آه



Before

after

AX	00	90
BX	01	FC

50	201FCH = [BX]
----	---------------

AX	00	50
BX	01	FC

90	201FCH = [BX]
----	---------------

12

LEA Instruction :-

LEA = Load Effective Address

It does not modify Flags

LEA BX, [DI]

ان معناها

اولاً في LEA لازم يكون جهة Source بينا قواله
مفروض ج ا , In هادي معناها حور محتويات DI و هادهم
في BX يعني !

LEA BX, [DI] = MOV BX, DI

بلا بساطة ! *

ex

MOV BX, 1000H

MOV SI, 2000H

LEA DI, [BX+SI]

معناها قيمة DI حتوي 3000h

after	
SI	2000h
DI	3000h

الفرق بين LEA وال MOV :-

1- LEA BX, [DI]

2- MOV BX, [SI]

في الحالة الاولى لو استخدمت LEA جيني ال processor وياخذ القيمة
التي في SI كل مايل من غير ان يكتب عنوان ال source وياخذها في ال BX
بينما في الحالة الثانية لو استخدمت MOV جيني ال processor وياخذ
كافة قيمة ال SI وياخذ ال physical address وياخذها في ال mem Location
ولاخذ محتويات ال Location وياخذها في ال BX

- LDS Instruction :-

قبل لما كنا نكتبوا في مكان عنوان بين قوسين [number] كان يؤثرنا على Location واحد في ال memory. طول ال 8bit ولو كان بي 16bit كنا نزيدوا في ال location واحد، اللف نفس الوقت، يعني :-

LDS BX, [4326H]

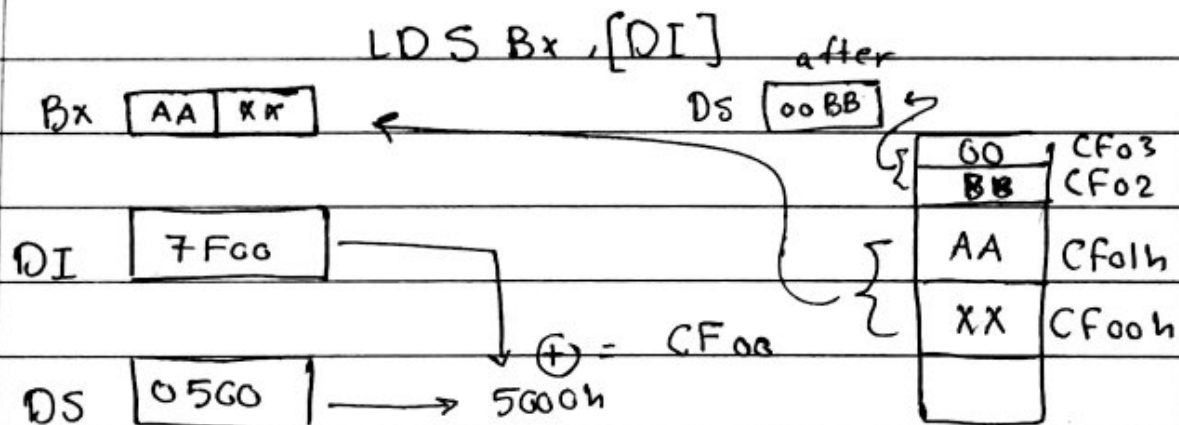
حنقل مكتوبات. العنوان اللي يؤثره 4326H إلى BL

ونقل مكتوبات العنوان 4327H إلى BH ونحوه.

الفرق هنا أن Data se حتمين ولها قيمت قيمة العناوين اللي يرميها

4328h & 4329h ex Assume DS = 0500h

DI = 7F00h



ex:- Assume DS = 500h, SI = 7D0h, Bx = 2Dh
determine the contents of all the affected registers if
the following instruction is executed :-

LDS DI, [Bx + SI]

$Bx + SI = 07D0 + 002D = 07FDh \Rightarrow 5000 + 07FDh$

physical address = 57FDh

DI [AA XX]

DS [00 11]

[14]

00	5800
11	57FF
AA	57FE
XX	57FD

$$DS = 1200 \rightarrow 12000$$

$$\text{physical address} = 12000 + 200 = 12200h$$

لا أنه ال DI طول 16bit منناها حناخذ مكتوبات

$$DI = 12200h \text{ و } 12201h \text{ الى صي تبدأ قيمة } DI = 12200h$$

وقلنا في LDS قيمة DS تنفي و حناخذ القيم

$$\text{القيمة التي بدنا حناخذها الى صي } 12202 \text{ و } 12203$$

$$DS = FFEH$$

—: LAHF :—

copies bits 0-7 of the Flags register into AH

ال In هاري ، لما تنفي نكتب برنامج وفي وحدة من ال Flags نكتب

LAHF « Load AH Flag » حناخذ قيمة AH الى 02

02 في

لأنه ال processor في In يتعامل في ال Flags بالكتابة الى

(CF) (OF) (PF) (CF) (AF) (OF) (ZF) (SF)

دائماً نتعامل مع أساس ال Flags كلهم إيهنا و حناخذ الهم الهم الهم الهم

$$\text{في ال AH هو } 00000010 = 02$$

دائماً لو كتبنا LAHF حناخذ قيمة AH الى 02 صي كانت

بالترتيب من الهم الهم الهم الهم الهم الهم الهم الهم الهم

نفس الهم الهم الهم الهم الهم الهم الهم الهم الهم

LAHF

بواحد

$$\text{MOV AH, } 0FFh \text{ or } \text{MOV AH, } 255$$

$$FF = \text{dec } 255$$

SAHF:-

قبل قلنا أنه LAHF لأن لقبها خيفرك قيمة AH وتلكها 02h دنا .

بأنه لو قبلها كنت ضايفك ان In هادي و SAHF من ريسر ؟

SAHF خدمتها أنه تخزن القيمة الحالية لـ AH ولو حي

بها In زي LAHF متيكونش AH لـ 02h لا ! شت ريسر
لنرض مثلاً التالي

MOV AX, 1234h →

12	34
----	----

SAHF → هذا معناها خزن 12

LAHF → AX

14	34
----	----

خزنه قيمته AH بمقدار 02h بجانب قبلها

SAHF , بي تفهم أكثر جربها كمان

((IN, OUT))

معنى الكلام هذا أنه سوف العنوان 2E من IN AL, 2EF

فيه وخو مكتوبات و دخله في AL يعني يحامل في Source لمليون

رغم على الأساس كائن عنوان

ex

MOV DX, 03F8H

IN AL, DX → اولاً رد بالك تقول انه In هادي

فدك لان size مش زي بعض الاللا , ان DX امني لقصد بيها

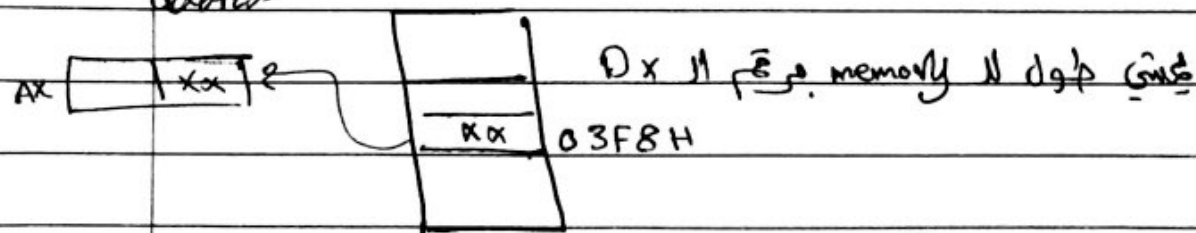
لأنها عنوان وتحتل location كـ 8-bits

* ملاحظة مهمة:

ان IN و OUT مضموش الا بال A واما AX او AL او AH

الباقي Error ..

جیال Dx عنوان ~~ت~~ فکرم فی ما یسب ال Physical address



1120

Out: -

ЕК

OUT 30H, AL

یہی خود مکتوبات AL پر ہے

في القسم الذي يأتيه $physical = 36$ أو

Before

after

AL	50H
30H	40H

AL $\boxed{50H}$, 30 $\boxed{50H}$

OUT DX, AX

توبه و توبه و توبه

Before

physical

$$A_x \quad \boxed{3050}$$

after

میں ہوا	۱
کے اہلی	۱

0x	2177h		2179h
	30	2178	45
	50	2177 ←	40

Regulation

Before

number 2901 91

181

ملاحظة هامة: في الـ IN يجب أن يكون «des = A» لما Ax
او AL او AH

في الـ OUT يجب ان يكون «Source = A» لما Ax او
AL او AH

~~6.13 am~~
~~4 Jan~~

19

Arithmetic Instructions :-

Addition, Subtraction, multiplication, Division.

Data Formats:-

Unsigned, signed « binary , bytes, words »

Unpacked decimal bytes

packed decimal bytes

ASCII numbers

Intel 8086 80 Instruction

ADD, ADC, INC, DAA, AAA \Rightarrow addition

SUB, SBB, DEC, DAS, AAS \Rightarrow Subtraction

MUL, IMUL, AAM \Rightarrow Multiply

DIV, IDIV, AAD \Rightarrow Division

يجب التركيز في مكتوبات الآلة قسم « لأنه مرات تبي MCQ »

Binary addition:-

Can add data directly From one memory location to another « X »

مرات يضيفك صاع ونزل في أسياد زي هكي!

تقدر اضيف بيانات في حالات زي هكي :-

ADD AL, BL
Reg \xleftarrow{L} Reg

ADD AL, 30h
Reg \xleftarrow{L} Im

ADD AX, [BX]
Reg \xleftarrow{L} memory

لازم تركيز في الجدول و بتجرب
كل شي شوي أمثلة بيش متناظرة :

Destination , Source

Register , Register

Register , memory

Memory, Register

Register , Immediate

Memory , Immediate

Accumulator, Immediate

Reg \xleftarrow{L} Reg
Ax « AH , AL »

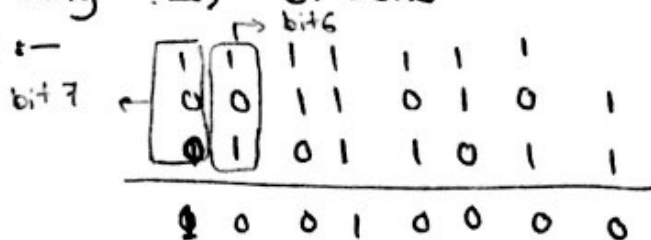
* مرات يضيفك في الذاكرة و يملك Seg
بمعبر مانشوف Seg في Add مينا Error

- can add Immediate data to Reg or memory "✓"
 - can add From a Reg to Reg "✓"
 - " " " a Reg to memory "✓" or memory to Reg "✓"
- فاهم الجدول ومجرب على كل المرح والخط بالأسفل!

- Overflow:-

- carry From bit 6 to bit 7 but no carry From bit 7 to carry \Rightarrow $OF = one$ $CF = zero$

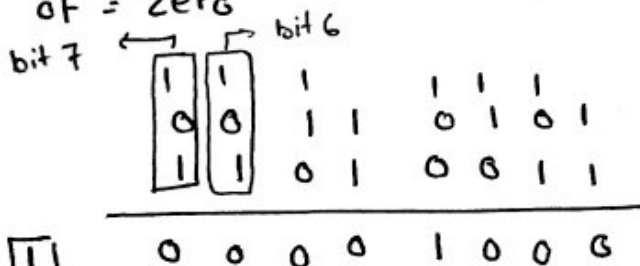
ex:-



- لاحظ هنا انه يوجد carry من bit 6 الى bit 7 ولكن لا يوجد carry في نهاية الناتج \Rightarrow $OF = 1$

- carry From bit 6 to bit 7 and carry From bit 7 to carry \Rightarrow $OF = zero$

ex:-

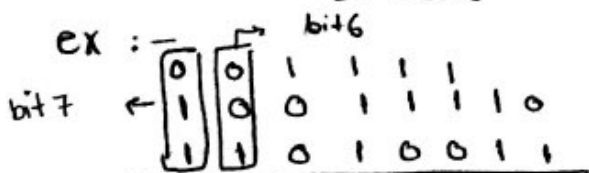


carry \rightarrow 1

- يوجد carry من bit 6 الى bit 7 ويوجد carry في نهاية الناتج اذا $OF = zero$, $CF = one$

- No carry From bit 6 to bit 7 but there is a carry From bit 7 to Carry $OF = one$ $CF = one$

ex:-



carry 1

- لاحظ هنا انه يوجد carry من bit 6 الى bit 7 ولكن لا يوجد carry في نهاية الناتج $CF = one$, $OF = one$

مرار بجيبك في الامتحان الحالات هاري في صورة! هن اوضح وخط !!

Examples:-

"Reg to Reg"

مثال حقة :-

MOV CL, 115d

MOV BL, 79d

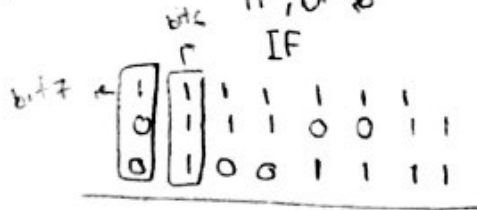
ADD CL, BL

ال status Flags يتأثروا لَمَّا ار control

TF, OF, IF

ZF, CF, CF

AF, PF, SF



No carry

OF = one

الناتج مثل بإشارة إذا سيكون على الشكل التالي

$$-128 + 64 + 2 = -62$$

$$CL = -62d$$

"Immediate to Reg"

MOV DL, 12H

ADD DL, 33H

جميع 12H إلى 33H ويضاف في DL

"Memory to Reg"

ADD AL, [DI]

or

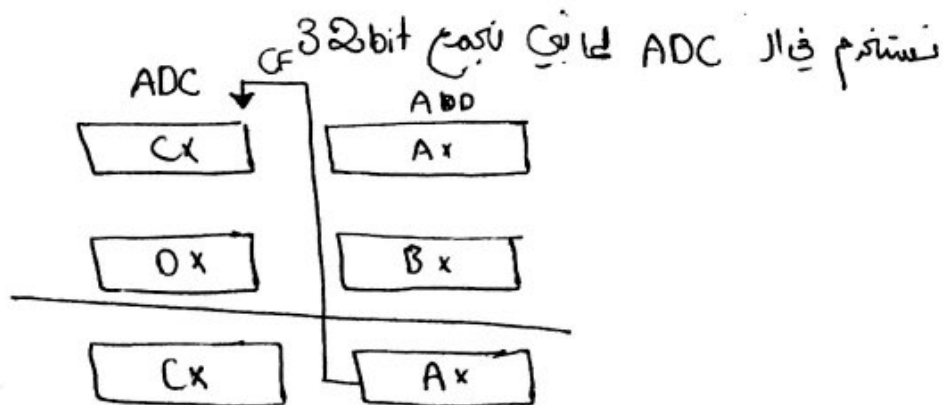
ADD BL, [CI]

وأيضا !

ADC => Add with Carry

يعني حتم مع و لو كان عندك no carry لا يضاف شيء. also

Destination + Source + Carry Flag => Destination



4

INC Destination :-

INC $\rightarrow +1$

DEC INC
مات يكتفي في اختار

operand \leftarrow INC

Carry Flag is not effected "✓"

The operand can be a Reg or memory or Location "✓"

ex

STC

CF=1

MOV AL, 5

INC \leftarrow carry

ADC AL, 1

8 = 1 + 1 + 1 + 5 = الناتج

INC AL

وكتا

SUB Instruction:-

SUB operand 1, operand 2

Destination

Source

Destination = Destination - Source

MOV AL, A7h \rightarrow 167d

MOV BL, 3Fh \rightarrow 63d

SUB AL, BL \rightarrow 167 - 63 = 104d \rightarrow AL = 68h

SUB AL, 5h \rightarrow 68h - 5h \Rightarrow AL = 63h

- subtract a 44H from 22H where 22H loaded into CX, the Sub Instruction using Imm 44H?

MOV CX, 22H

SUB CX, 44H \rightarrow CX = -22H

- Can sub imm data from a Reg or memory
- " sub a Reg from a Reg
- Can sub a Reg from memory
- Can sub memory from a Reg
- Can NOT sub data in one memory location from that in another memory location.

يأبوا في اختار و X و ✓

It subtracts a byte to byte or a word to word .. ✓ ..

It effects AF, CF, OF, PF, SF, ZF Flags .. ✓ ..

لو قالك DF او TF مناهل "X" حة IF

For subtraction CF acts as borrow Flag .. ✓ ..

- Execution of SUB instruction:-

Take the 2's complement of the subtrahend

يخاف ال source من ال 2's

Add it to the minuend

واضيف ال 2's لل Dest

ex

MOV AL, 8FH → MOV BH, 23H → SUB AL, BH

AL = 0011 1111

AL = 0011 1111

BH = 0010 0011

- BH => - 0010 0011

CF = zero → borrow في ال carry

ZF = zero

PF = zero

SF = zero

↓
 0011 1111
 + 1101 1101
 1 0001 1100

If the CF = 0 the result is positive and the destination has the result

If the CF = 1 the result is negative and the destination has the 2's complement

ex

MOV AL, 4Ch

4C
- 6E

MOV BL, 6Eh

- 22h

SUB AL, BL

0100 1100

0100 1100

- 0110 1110

= + 1001 0010

CF = 1 ← فلا حة هنا ان SF = 1 اذا ال عدد سالب

فلا حة هنا ان SF أكبر من 4C
 فالكه سيكون الناتج بالسالب
 نأول ال Binary ونعشوف من يمين

6

SBB « Subtract With Borrow »

Destination = Destination - Source - Carry Flag

If CF = 0, SBB works exactly like SUB. ✓

If CF = 1, SBB subtract one from result. ✓

Ex « Analyze the following program :-

DATA_A DD 62562FA H

DATA_B DD 412963B H

RESULT DD ?

- 1- MOV AX, WORD PTR DATA_A \Rightarrow Ax = 62FA H
- 2- ~~MOV~~ SUB AX, WORD PTR DATA_B \Rightarrow Ax = 963B H
- 3- MOV WORD PTR RESULT, Ax \Rightarrow Ax \Rightarrow خزن الـ Ax
- 4- MOV AX, WORD PTR DATA_A + 2 \Rightarrow Ax = 0625H
- 5- SBB AX, WORD PTR DATA_B + 2 \Rightarrow SUB 0412 with B0
- 6- MOV WORD PTR RESULT + 2, Ax \Rightarrow خزن الناتج

بين متناظير لما يلي تاخذ بيانات من الذاكرة A او الذاكرة B
رتبهم كالهم في جدول Low ثم high

الذاكرة A \Leftarrow

06	\rightarrow هذا الرقم 3
25	\rightarrow هذا الرقم 2
62	\rightarrow هذا الرقم 1
FA	\rightarrow هذا الرقم 0

الذاكرة B \Rightarrow

04
12
96
3B

في الخطوة الأولى $62FA = Ax$ في الـ Ax في الـ 62FA في الخطوة الثانية المخرج من الـ Ax 963BH

وخرن في الـ Ax يعني $62FA - 963B = CCBF$ وكد $CF = 1$
في الخطوة الثالثة قالك حط مكتوبات الـ Ax في الذاكرة RESULT يعني

في الخطوة الرابعة قالك حط $0625H$ في الـ Ax وفي الخامسة قالك المخرج من الـ Ax 0412 ومناه الـ Borrow اللي هو "1" لانه الـ CF من فوق كان 1

يبدا ناتج Ax = 0212H

في الخطوة السادسة قالك حط مكتوبات الـ Ax في RESULT + 2 اللي هو تبدا الذاكرة RESULT من هنا :-

الذاكرة RESULT \Leftarrow

02
12
CC
BF

• φ and ψ , one operand is

It decrement the byte or word by one "✓"

The DEC instruction subtract 1 from any register or memory location or segment register "X".

↳ The DEC instruction subtract 1 from any register or memory location except a segment register "✓".

The operand can be a register or memory location. ✓

CF is not effected „√“

NEG Destination :-

كأنفا ۛ ۛ الرّمم NEG →

It modifies Flags AF, CF, OF, PF, SF, ZF

CMP Des, Src "Comparison"

انظر الجزء ١٧٤

أو CMP متغيرتي قيمته أو Des ولا Src الكمية الموصية التي تغيرها
أو Flags

The value of Source and Destination does not change but Flags are modified to indicate the result. ✓

از CMP یوضاحت هل از Des انکبر او ا هفراو یسادی Src
و هفراکه طریق از CF و ZF

CF	ZF	
0	0	$\Rightarrow \text{Des} > \text{Src}$
0	1	$\Rightarrow \text{Des} = \text{Src}$
1	0	$\Rightarrow \text{Des} < \text{Src}$

- The instruction CMP to compare source and destination operands it performs :-

a) Addition **b) subtraction** e) division d) multiplication

- During comparison operation the result of comparing or subtraction is stored in :-

a) memory b) registers c) stack **d) nowhere**

لا يوجد نتيجة الـ CMP في مكان ما

- In general, the Source operand of an instruction can be :-

a) memory location b) register c) immediate data
d) all the above

- The instruction that enables subtraction with borrow is:-

a) DEC b) SUB **c) SBB** d) None of the mentioned

MUL Source

IF source operand is a byte $AX = AL * \text{Source}$

IF source operand is a word $(DX:AX) = AX * \text{Source}$

Source operand cannot be an Immediate data "✓"

Flags affected by MUL are CF, OF
 MUL always multiplies in the A register
 If the source operand is a word, the multiplier is in the DX register
 After MUL: —

CF/OFF = 0 IF the upper half of the result is zero
 يعني بيش نقول أنه إذا CF و OF = 0 فهذا يعني أن الـ AH يكون كله صفراً
 في حالة الـ byte ولو كان word لا يهم لأن الـ DX كله صفراً
 لو كان أي شيء غير الصفراً، معناه كذا CF و OF

ex

```
MOV AL, 5H
MOV BL, 10H
MUL BL
```

يعني اضرب ما مكتوب في الـ BL في الـ AL وخرن
 في الـ AX لأنه Byte
 ولو كان الـ AH مش صفراً معناه في
 CF و OF

5H = 5d
 10H = 16d
 $5 \times 16 = 80d$
 $80d = 50H$
 $AX = 0050H$

CF = zero, OF = zero

ex

```
MOV AX, 2000H
MOV BX, 0100H
MUL BX
```

يعني اضرب ما مكتوب في الـ BX في الـ AX و
 خزنه في الـ AX الـ Low و الـ DX الـ High
 ولو كان الـ DX مش كله صفراً معناه
 في CF و OF

2000h = 8192d
 0100h = 256d

$8192 \times 256 = 00200000h$

$AX = 0000h, DX = 0020h, CF = 1, OF = 1$

Exampels :-

1) MOV AL, 80h \Rightarrow MOV BL, FFh \Rightarrow MUL BL

AX = 7F80h

AH = FFh, AL = 80h

لَمْ يَكُنْ فِيهَا
CF/OF = 1

2) MOV AX, 0001h \Rightarrow MOV BX, FFFFh \Rightarrow MUL BX

AX = FFFFh, DX = 0000h

CF/OF = zero لَمْ يَكُنْ فِيهَا

3) MOV AX, FFFFh \Rightarrow MOV BX, FFFFh \Rightarrow MUL BX

AX = 0001h, DX = FFFFh

CF/OF = 1 لَمْ يَكُنْ فِيهَا

4) MOV AX, 0FFFh \Rightarrow MOV BX, 0FFFh \Rightarrow MUL BX

AX = E001h, DX = 00FFh

CF/OF = 1 لَمْ يَكُنْ فِيهَا

وهكذا !!
" انظر الجدول 191 " مهم

" IMUL Source "

يعني الضرب بالشارة !

في الـ IMUL لو كان الناتج كدمومب مفروض الـ upper يكون كله

فيها، بيش نقول انه CF/OF = 0

يعني لو كان Byte المفروض AH تكون فيها، بما ان كدمومب

ولو كان word DX = 0

لو كان العدد الـ upper يكون واحدات !

examples:-

1) MOV AL, 48d
MOV BL, 4d
IMUL BL

$$48 \times 4 = 192d = C0$$

$$Ax = 00C0$$

$$Ax = 0000\ 0000\ 1100\ 0000$$

مثل ياشارة

لانه البت الاخير 1 فالفرفف بيض نقول انه ماغيث 0F لازم يكون

AH=FF , واهني بيوي 00 معناها 0F=1

2) MOV AX, 48d
MOV BX, 4d
IMUL BX

$$48 \times 4 = 192d = 00C0$$

عنا انت word معناها سينتزن

في AX و BX

0F=0 هنا ان

0F=0 هنا ان

0F=0 هنا ان

0F=0 هنا ان

$$\begin{array}{r} 0000\ 00C0 \\ \downarrow \quad \downarrow \\ 0x \quad Ax \end{array}$$

قارن بين مثال "1" و "2" !!

3) MOV AL, -4d
MOV BL, 4d
IMUL BL

$$4 \times -4 = -16d$$

الناتج سينتزن في Ax

$$16 = 0000\ 0000\ 0001\ 0000$$

$$-16 = 1111\ 1111\ 1111\ 0000$$

AH

AL

نلاحظ ان العدد في AL مثل بالاشارة السالبة لانه البت الاخير 1

اذا لو اردنا القول ان 0F=0 يجب ان ماكوي AH=FF وهذا الشرط قد تحقق

so 0F = zero

1- MOV AL, 80h \Rightarrow MOV BL, FFh \Rightarrow IMUL BL

تذكر أن الأرقام في حالات IMUL تمثل بالشارحة

AL = 80h \Rightarrow العدد سالب \rightarrow 1 000 0000

AL = 80h \Rightarrow تأخذ 2^س

BL = FFh \Rightarrow العدد سالب \rightarrow 1 111 1111

تأخذ 2^س 0000 0001

BL = 01

$$AL * BL = 0080 = \underbrace{0000\ 0000\ 1000\ 0000}_{AH} \underbrace{0000\ 0000}_{AL}$$

العدد في AL هو سالب ولكن AH \neq FF إذا CF = 1

2- MOV AX, 0001h \Rightarrow MOV BX, FFFFh \Rightarrow IMUL BX

AX = 0001 \rightarrow العدد موجب له نقوم بأخذ 2^س

BX = FFFF \rightarrow العدد سالب تأخذ 2^س

BX = 0001h

بما أنه BX سالب و AX موجب فإن حامل الضرب سيكون سالباً
عند عدد سالب فيجب أخذ 2^س !

$$DX\ AX = \underbrace{0000\ 0000\ 0000\ 0000}_{DX} \underbrace{0000\ 0000\ 0000\ 0001}_{AX}$$

نقوم بأخذ 2^س

DX AX =

$$\underbrace{FFFF}_{DX} \underbrace{FFFF}_{AX}$$

العدد في AX هو سالب و ال DX مملوءة ب FFFFh

ان F = zero

زيادة التوضيح أنظر ص 195 !

—: DIV/ IDIV : Unsigned / Signed Division :—

أنتظر البعول في مبر 800
في المبر الـ 800 عند القسمة له يكون هناك CF لأن عند قسمة عدد على عدد
سيكون الناتج عدد أكبر من 255 !

عند الضرب في حالة Byte كان AH أو upper وكان يحدد ال CF & OF
في DIV يوضع باقي القسمة في AH
وفي حالة word يوضع باقي القسمة في DX

For ex

MOV AX, 0083H
MOV BL, 02H
DIV BL

83h = 131d
2h = 2d

$$\begin{array}{r} 65 \\ 2 \overline{) 131} \\ \underline{130} \\ 001 \end{array}$$

الباقي 1
والعدد 65d

AH يوضع الباقي في
والعدد في AL

Ax = 0141h

MOV DX, 00
MOV AX, 8003H
MOV CX, 100H
DIV CX

8003H = 32771
0100H = 256

$$\begin{array}{r} 128 \\ 256 \overline{) 32771} \\ \underline{32768} \\ 00003 \end{array}$$

الباقي في DX = 0003h
العدد في AX = 80h = 128d

DX AX = 0003 0080

وهكذا

IDIV ،، القسمة بالشارع ،،

perform signed division operation:-

If source operand is a byte $\Rightarrow AL = AX / \text{source}$

If source operand is a word $\Rightarrow Ax = (Dx Ax) / \text{Source}$

11 Source operands can not be an Immediate »

أنظر جدول 803 ⇒ مكتوباته قد تأتي في صورة ما يلي هذا ال Instruction ١٠

That ~~all~~ all arithmetic instructions of 8086 are for integer numbers, For Real numbers (5.32) -----, coprocessor is used.

a) 8088

b) 8087

c) 8684

لهذا الـ Processor (APU)
لذا قام الـ بنمونه على فوائده وليس
أعدادها صافية!

في ال IDIV زي ما قلنا هنوري نكو ال word ل byte ، وال word ل double word
 لازم بيبي دايماً نضمن انه البت يكون أكبر من الحقار لازم ال processor متكاملين مع ال
 بالوالم ، فلهذا هنستخدم two instruction يكون ال word → DW
 Byte → B

EBW \rightarrow convert Byte to word

حسبوف الرقم التي كنهه اللزيتونه من حيت ويسوف البيت للاضر مناع الاشارة
 بواكان 1 يعي حيت الجدر بالواكه وكلوكان البيت للاضر مناع الاشارة
 يعي الـ 8 حيت الجدر بالاضر

Ex

MOV AL, -48 \Rightarrow

$$48 = 00110000$$
$$-48 = 11010000$$
$$-48 = 00$$

١١٥١ ٥٥٥٥
 CBW
 إلى هذا البيت الأخير

$$Ax = \begin{pmatrix} 1111 & 1111 & 1101 & 0000 \end{pmatrix}$$
$$A \times = FF D 0$$

$$CWD \Rightarrow ex \quad Ax = -3241$$

$$Ax = 1111001101010111$$

$$DX Ax = 1111111111111111001101010111$$

$$DX Ax = F537$$

$$\begin{array}{c} Dx \\ \downarrow \\ FFFF \end{array}$$

$$\begin{array}{c} Ax \\ \downarrow \end{array}$$

المتحويل أو التمديد

مستفاد من الإخراج نواتج صحيحة عند القسمة مباشرة!

انظر المثال م 204 لمعرفة الفرق.

.. البرامج الموجودة في صفحات م 210+211 مهمة لـ آ ..

BIT MANIPULATION INSTRUCTIONS :-

Logical instructions " NOT, AND, OR, XOR, TEST "

Shift instructions " [SHL / SAL], [SHR / SAR] "

Rotate instructions " ROL, ROR, RCL, RCR "

The Logical instruction:-

AND, OR, XOR \Rightarrow D, S فيهم

NOT \Rightarrow فيها D فقط

جدول التوزيع لا Logical \Rightarrow 210 مهم
لازم تركيز شن صيا العاجبات التي نقدر نستخدمها Source
و ركز في الـ NOT حيث Destination فيها نوعيه فقط
يا! ما Reg or memory

$$x \oplus 1 = \bar{x}$$

$$x \oplus 0 = x$$

AND, XOR, OR \Rightarrow Flags فيهم يتأثر إلا AF

NOT \Rightarrow Flags فيهم
ميتا تروشن

حتى الـ AND, XOR, OR الـ Des فيهم يكون يا! ما Reg او mem
والـ Source Reg او mem او Imm

- * ----- clears bits of a binary number
- a) OR b) AND c) XOR
- نستعمل توضيح ذلك لاحقاً

* AND clears bits of a binary number called masking ✓

* In 8086, the AND instruction often executes in about a micro second ✓

Important:-

** AND uses any mode except memory to memory and segment register addressing.

يعني الـ AND تقسم مع mode، إلا mem to mem او Seg، رديك في النقطة هاري!
تأني مع او فقط

- * An ASCII number can be converted to BCD by using AND to mask off the left four binary bit position ✓

ضروری ترکیبی! استعمال - AND

من معنی اللام هذا؟!
 اولاً ال mask یقصد بیجا انہما تستعمل مع ال processor تجربہ
 پہلے رقم مکین اوٹسی انت تبیہ
 نفرض انہ کان کدی رقم فی ال ASCII = 3135
 بیہ بار BCD فدیہ AND مع 0F0F

MOV BX, 3135H
 AND BX, 0F0FH => ~~3135H~~

Note:-

After AND:-

CF and OF become zero

PF and SF and ZF are upted

رکز آہنی مکیناں سیو AF

لو دلیب منک انک تاکولہ حرف صغیر یعنی b الی B تقد، تستعمل
 AND کیف!

b = 62H

استعمل AND DF

MOV AL, 'b'

AND AL, DF

حتیٰ حد B ← 42

$$\begin{array}{r}
 01100101 \\
 \cdot 11011111 \\
 \hline
 01000010
 \end{array}$$

معناها لما بتی تاکول حرف صغیر الی حرف کبیر استعمال AND DF

مرات يقولك ~~معا~~ خبي البت الرابع يطبع من لاي ارقام كان !
شن حذير؟

البتات الباقيات نخليهم 1 اذ لا نبيح من ندين صفر ، يعني :-

لنرفس الرقم كان 35 وقالك خبي البت البتات من
يعني مفوض نيرك AND مع $\begin{array}{c} 1111011 \\ \downarrow \quad \downarrow \\ F \quad B \end{array}$

وهكذا لانه اي ارقام AND مع 0 = 0

* لماتبدا زابط استخدمات ال AND تعرف لما يقولك اكتب برنامج هل مستخدم
AND او لا !!

— : OR : —

Destination and source can not be both memory locations ✓

ال Flags يثاودا وال $Zero = CF = OF$

* OR 1 مع اي شي يدطي 1
* OR 0 مع اي شي يدطي الشيء نفسه

xxxx	xxxx	
OR	0000	1111
<hr/>		
xxxx	1111	Result

* كان طلب منك بت معين لازم يكون واحد المفوض تفكيرك حول يعني لا OR
لانه OR مع 1 للرقم حيطي 1
مثلا كان عندك الرقم 35 وقالك نبيك تاكولي البت الخامس
والا ارس فيرك OR مع 30H

ركز! كان بي بت معين من اعرف انك مستعمل AND
OR " " " واحد " " " " " " " "

لوبي تقول من حرف كبير الى حرف صغير استخدم $OR\ 20H$

مثال :- $MOVAL, 'A' \Rightarrow AL = 41$

$OR\ AL, 20H \Rightarrow AL = 61$

$41 \Rightarrow$

0100 0001

$20 \Rightarrow$

0010 0000

0110 0001

↓ ↓
6 1

يعني من كبير الى صغير $OR\ 20H$
من صغير الى كبير $AND\ DF$

— : XOR : —

المعلومات المهمة التي تألمت عليها في الـ OR هي نفسها في الـ XOR

$$\begin{array}{r} \textcircled{+} \quad \begin{array}{cccc} XXXX & XXXX \\ 0000 & 1111 \\ \hline XXXX & \bar{X} \bar{X} \bar{X} \bar{X} \end{array} \end{array}$$

مناها لوبي تنفي سبي مابين تقدر اديره $XOR\ FF$ جيد طيك الـ NOT متاكده !

$NOT\ AL = XOR\ AL, FFH$

$AND\ AL, 00H = XOR\ AL, AL$

كان بي ا دخل حرف صغير او كبير وتبيح يطبع العكس متاكده
يعني كان كبير يطبع صغير وكان صغير يطبع كبير ريس

XOR 20H

NOT :-

- * The operand can be a register or memory location ✓
- * NOT can use any addressing mode except segment register

« الفرق بين NOT و NEG »

The NOT Function is considered logical, NEG Function is considered an arithmetic operation.

ار آتو متاثرش كل ال Flags ! طلاقاً

ار NEG تاثر

Ex

The one's complement of FOH is OF

MOV AL, FOH

NOT AL \Rightarrow AL = OF

1111 0000

NOT \Rightarrow $\begin{array}{cc} 0000 & 1111 \\ \downarrow & \downarrow \\ 0 & F \end{array}$

Examples:-

تبع البرنامج !

MOV AL, 55H \Rightarrow AL = 01010101

AND AL, 1FH \Rightarrow $\begin{array}{r} 01010101 \\ 00011111 \\ \hline 00010101 \end{array} \Rightarrow$ AL = 15H

OR AL, 0FH \Rightarrow $\begin{array}{r} 00010101 \\ 11000000 \\ \hline 11010101 \end{array} \Rightarrow$ AL = 05H

XOR AL, 0F

$\begin{array}{r} 11010101 \\ 00001111 \\ \hline 11011010 \end{array} \Rightarrow$ AL = DAH

NOT AL

\rightarrow AL = 0100101 = 25H

!!

- * Mask : clear a bit to zero ✓
- * To clear a bit we AND it with zero ~~AND~~ AND with 1 ✓
- * Set : setting a bit to 1 : OR with logic 1 ✓
- * Toggle : XOR to reverse the logic level ✓

- change bit(2) (D3) in BL to the opposite value and all other bits would remain unchanged ?

بما اننا قاي غير البت الثاني من اي رقم كان واني مغرقت البت
التالي في الرقم هل هو واحد او صفر فندير XOR
قاي البت الثاني ← 0000 0100

XOR 04H

*** Clear a general purpose register using a four different instructions ?

قال clear يعني صفرهم وكل مرة استخدم! يعني شكل!

MOV AX, 0 → ① ✓

SUB BX, BX → ② ✓

XOR CX, CX → ③ ✓

AND DX, 0 → ④ ✓

- * To change that letter to uppercase, we subtract 20H "✓"
- * To " " " " lowercase " add 20H "✓"

اكثر اكر اكثر !!!

عندك اكثر من طريقة تقول ليها هي الحرف
" XOR, ADD, SUB, AND, OR "

TEST " Logical compare Instruction "

The TEST instruction performs the AND operation, The difference is that the AND instruction changes the Des operand, while the TEST instruction dose not, (A TEST affects only condition Flags)

The TEST instruction Function in the same manner as a CMP instruction, The difference is that the TEST instruction normally tests a single bit, while the CMP In tests the entire byte or word.

TEST DL, DH => DL is ANDed with DH
لكنه مينز نشي

MOV AL,

Test AL, 1 =>

لو ا هيج ZF=1 فهذا

يعني ان الدير زوجي

ex

MOV AL, 31H

Test AL, 1

لو ا هيج ZF=0 فهذا

يعني ان الدير فردي

31 =>

0011 0001
• 0000 0001

0000 0000

ZF = zero => الدير فردي

" Test 1 " يعني لما بي نعرف هل الدير فردي او زوجي وسوف يشي في ZF

- باصلي لو بي نعرف هل العدد سالب او موجب اللي هو انه طريق البت الاخير
فندير Test 128 لو $ZF=1$ موجب لو $ZF=0$ منهاها سالب

MOV AL, -4

Test AL, 128

AL = 8 bit

4 = 0000 0100

-4 = 1111 1100

AL = ☐ 1

Test 128

1111 1100

• 1000 0000

1000 0000

ZF = 0

معناها العدد اللي دخلته سالب ✓
وهذا الرجاء التركيز فقط

#

سيتم إلحاق باقي المنهج الـ داخل في الدفتر
التالي قبل الامتحان بكون الله .

Shift and Rotate Instructions:-

Shift
 ↳ Logical «unsigned» (SHL, SHR)
 ↳ Arithmetic «signed» (SAL, SAR)

* Shift and Rotate instructions manipulate binary numbers at the binary bit level. ✓

Shift و Rotate لا يغيران ترتيب البتات Binary bits وإنما يغيران موقعها

Shift : position or move numbers to the left or right within Reg or memory

* The microprocessor instruction set contains Four different shift instructions.

SHL, SHR, SAL, SAR
 ← ٤ مرات يقولون أو

two or

three

ويعطونك ! كتر

↳ two logical shift

↳ two Arithmetic shift

الرجاء التركيز في المعلومات التي تكوني على * لأن قد تأتي MCQ أو ✓ و X

* Logical shift Function with unsigned numbers

* Arithmetic shift Function with signed numbers

* Logical shifts move zero in the rightmost bit For a Logical:

a) shift left

b) shift Right

معنى ما هنا ! نضيف من اليمين في الرقم معناها قاس نترك في الرقم لليسار

Shift left يمين

* zero to the Leftmost bit position For a Logical:

a) Right shift

b) Left shift

** Shifting means to move bits right and Left inside an operand

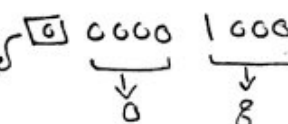
** All Four shifting instruction affecting the overflow and Carry Flag
يعني ان shift تأثر على CF و OF

A shift left always multiplies by 2 for each bit position shifted.
يعني كل shift left و حدة تكاثر ضرب الرقم في 2
مثلا!

كان الرقم عندنا 04 الذي هو 0000 0100 حيث في AL
MOV AL, 04h
SHL AL, 1

ولما ار shift الوبت كانها ضرب في اثنين معناها الناتج معوض يكون 08h! صاي نشوفوا:

AL = 0000 0100

SHL, 1 = 
CF

AL = 08h ✓

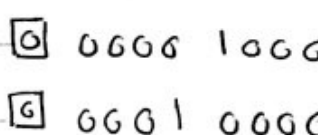
MOV AL, 04h

SHL AL, 2

اذ كان

الضرب = SHL = معناها $04 \times 2 = 08$ $08 \times 2 = 10h$ كأنه الضرب
في 2 مرتين لانه قالك 2, خلي نجرب ار SHL ونشوف اللام مع اولاه!!!

AL = 0000 0100

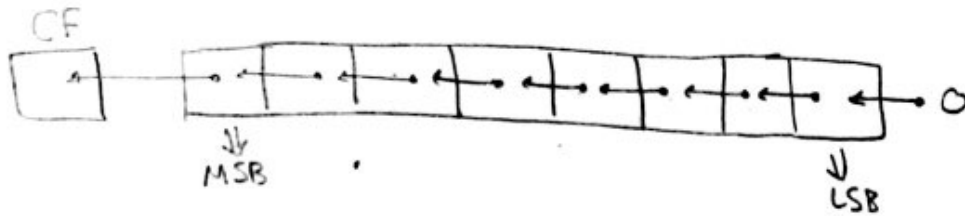
SHL AL, 2 = 
مرتين (ان شاء الله)

AL = 10h ✓

وهذا SHL
انظر في الدرس متاعها.

* During the shift operation, The MSB of the destination is shifted into CF and zero is shifted into the LSB of the destination.
 له التي هو البت الاخير او البت الاول
 البت الاول

بمعنى قال ان SHL محتوي البت MSB في ال CF والبت LSB دائماً وفي SHL سيكون zero لانه في البت في معرمة البت



- * operand count "Source" can be either an Imm or Reg CL
- * Des can be Reg or memory location. Reg C له البت في ال C
- * It modifies Flags CF, OF, PF, SF, ZF لا حظ في كرتش AF لانه
- *** OF = zero if the first operand keeps original sign. مليش في كرتش!

Examples: —

- MOV AX, BB17H \Rightarrow SHL AX, 1

AX = 1011 1011 0001 0111 Before

SHL, 1 [1] 0111 0110 0010 1110 After

762E = 2h x BB17h - بانه

- MOV DH, 6h \Rightarrow MOV CL, 4h \Rightarrow SHL DH, CL

قلا SHL كانه ضرب في 2 و اللف اربع مرات يعني

الناج معروف يكون DH = 60h

DH = 0000 0110

SHL, 1 = [0] 0000 1100

SHL, 1 = [0] 0001 1000

SHL, 1 = [0] 0011 0000

SHL, 1 = [0] 0110 0000

CF = zero , DH = 60h ✓

4 مرات

4

SHL = SAL

Shift Left = Shift Arithmetic Left

نفس المخرجات بالترتيب !

- MOV BL, F0H

SAL BL, 1

BL = 1111 0000

SHL, 1 = 1110 0000

Before \Rightarrow BL = F0H = -16 decimal

after \Rightarrow BL = E0H, -32 decimal \Rightarrow CF = one

نرجع نقول SHL و SAL كأنهم فرق في 2

- MOV AL, E0H

SAL AL, 2

AL = 1110 0000

SHL, 1 = 1100 0000

SHL, 1 = 1000 0000

Before AL = E0H = -32d

After AL = 80H = -128d

وهذا

Shift Right :-

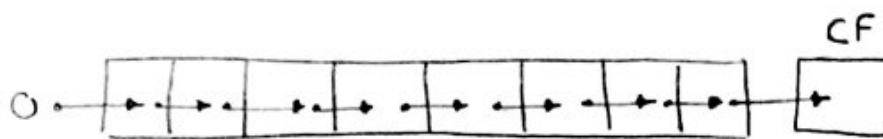
A SHR always divides by 2 For each position.

SHL II $\frac{1}{2}$ SHR II

SHR II و SHR I أنشأ قسم

- * During the shift operation, the LSB of the destination is shifted into CF and zero is shifted into the MSB of the destination.

SHL units, SHL units, SHL units



SHR \rightarrow 2 جہات

- ```
MOV DL, 40H
SHR DL, 3
```

DL = 0100 0000   

$$SHR,1 = 0010 \ 0000 \ \boxed{0}$$
$$\delta H R, 1 = 0001 \ 0000 \ \boxed{01}$$

SHR, I = 0000 1000 15

$$DL = 0.8h, CF = \text{zero}$$

- ```
- MOV AL, 9AH
  MOV CL, 3
  SHR AL, CL
```

$$AL = 1001 \ 1010 \ \square$$

SHR, 1 = 01 00 11 01 0

$$\delta H R_1 = 0010 \cdot 0110 \quad \square$$

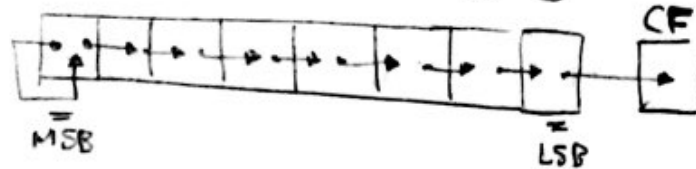
SHR, 1 = 0001 00 11 [6]

$$AL = 13h \quad , \quad CF = \text{zero}$$

SAR " Shift arithmetic Right "

* The LSB of the destination is shifted into CF and the MSB of the destination remains the same.

بمعنى :-
أنه (SAR) البت الأخير الذي هو ال MSB حقيقه يكرر نفسه عدد البتات التي تم نقلها
وكل مرة يسير فيها Shift ال Carry Flag حتى ياتي ال LSB ، كسيف !



الاختلاف بينه وبين ال SHR انه :
SHR -> إدخال صفر من اليمين
SAR -> إدخال البت ال MSB من اليمين
التي هي ال MSB من ال MSB

Ex

MOV AX, BB17H

SAR AX, 1H

Ax =

1	0	1	1	1	0	1	1	0	0	0	1	0	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

CF

 Before

SAR, 1

1	1	0	1	1	1	0	1	1	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

CF

 1

After SAR, 1 \Rightarrow Ax = DD8BH

لزيادة التوضيح انظر المثالين 229

7

—: Rotate Instructions :—

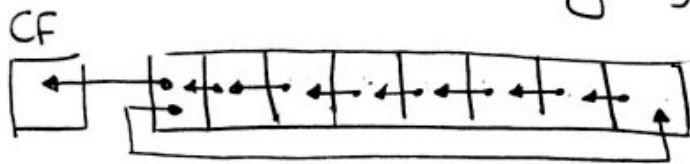
انواعها :-
 Rotate left (ROL)
 Rotate Right (ROR)
 Rotate Left through carry (RCL)
 Rotate Right through carry (RCR)

Destination → Reg
 → memory

ROL « Rotate Left »

- * The MSB of the destination is shifted into CF, it also goes to the LSB of the destination

بمعنى ان ROL خاوية ال MSB منقولة لـ CF و منقولة لـ LSB
 Carry Flag و LSB



الرجاء التركيز!!

- * Operand count can be either an Immediate data or Reg CL
- * It modifies Flags CF, OF, OF = zero if the first operand keeps original sign

MOV BH, 3Fh

ROL BH, 4h

Before BH = 0011 1111

After ROL, 1 = 0111 1110

After ROL, 1 = 1111 1100

After ROL, 1 = 1111 1001

After ROL, 1 = 1111 0011

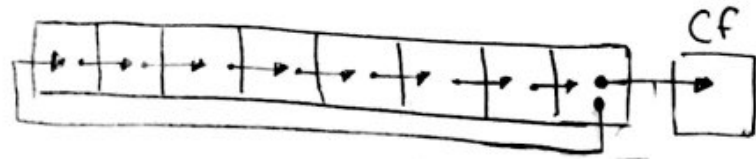
BH = F3h, CF = 1, OF = 1

8

ROR « Rotate Right »

رُكز ار رُكز ROR

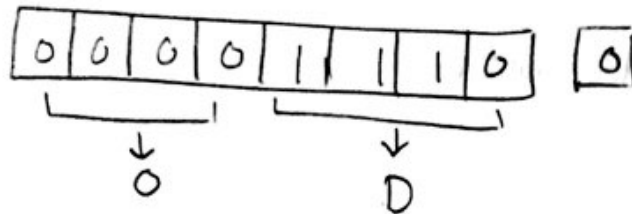
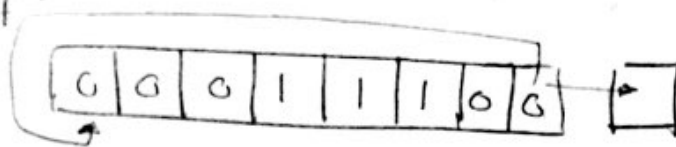
* The LSB of the destination is shifted into CF, it also goes to the MSB of the destination.



MOV AL, 1CH

رُكز ار رُكز ROL

ROR AL, 1



CF = 0
OF = 0

zero لانه
الدر قد مضى
على شئ
المحولة

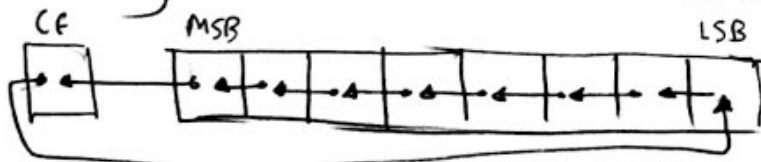
وهذا لانه

RCL « with carry »

رُكز في الذاكرة مع النجاة

* The MSB of the destination is shifted into CF, the old CF value goes to the LSB of the destination.

يعني المرة هار ار Carry مع لانه
ونجس في ار LSB و ار MSB حامي Carry



Ex

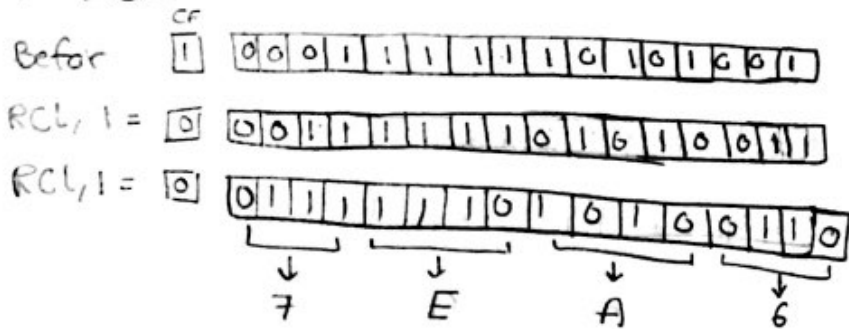
assume CF = one

9

MOV AX, 1FAGH

MOV CL, 2H

RCL AX, CL

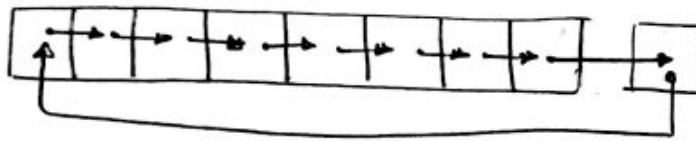


After RCL, 2 => AX = 7EAGH

RCL "Rotate ~~Left~~ Right with carry"

RCL note

- * The LSB of the destination is shifted into CF, the old CF value goes to the MSB of the destination.



تو ۲۳۶ تا ۲۳۱ به این مناسبت
ممنون از توجه و زبانه زدن شما!

#

XCHG AX = ROL AX, 8

ال Jump لها operand واحد فقط ، يعني تأجيل « \square Jump »
 فنقوم ال Jump في كتابة برامج اختبار طوية نوناً ما ، وتأكيدي ~~هذه البرامج~~
 على بعض الشروط فنقوم باستخدام ال Jump ، ما يعتبر مثال لها هو
 "if" في لغة C++ ! إذاً هذا !!

ال Jump لا لا يؤثر على ال Flags

operands:

- 1) short - Label \Rightarrow Jump short-xx
- 2) Near - Label
- 3) Far - Label
- 4) memory PTR [16bit] \rightarrow Near
 Error لم لازم 16 كنهنا يسطح Error
 يعني مش 8 bit
- 5) memory PTR [32bit] \rightarrow Far
 ال Far بيده علينا وطلع من ال Seg
- 6) Reg PTR [16bit]

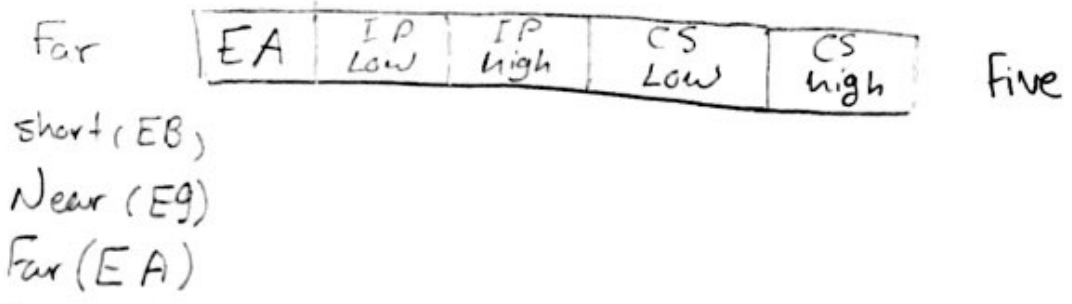
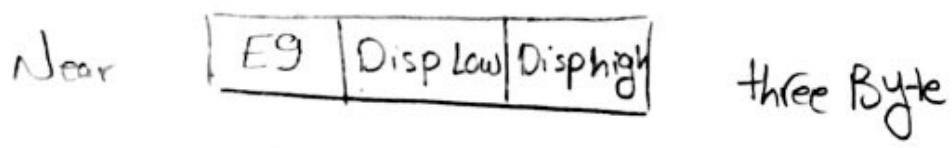
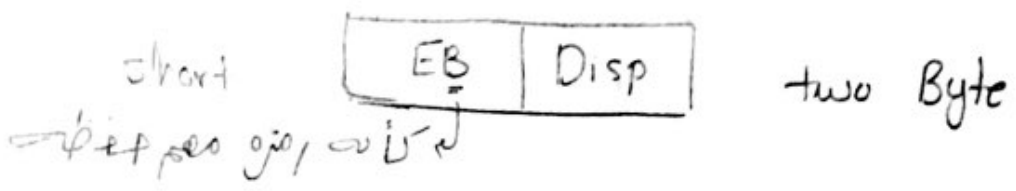
« أن في أنواع ال operands »

- Short jump: is a two byte instruction , within $+127 \rightarrow -128$
- Near jump: is a three byte " , within $\pm 32K$ bytes from the instruction in the current code segment
- Far jump: is a five byte , allows a jump to any memory location within the real memory system.

* The short and near jumps are often called:-

a) intrasegment

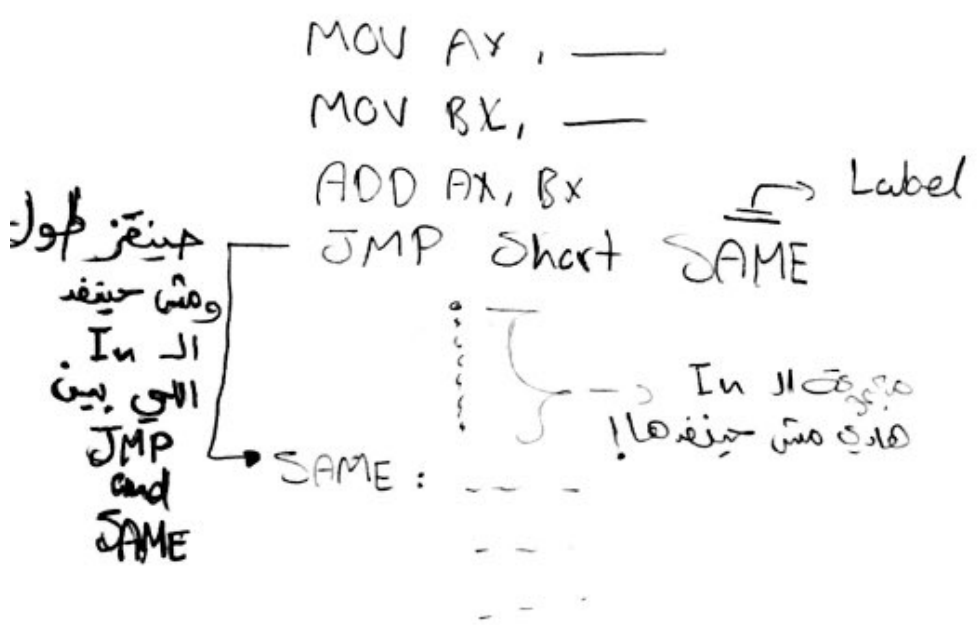
b) inter segment



• Called relative jumps because they can be moved with related software to any location in the current code Seg without change:-

- a) short jmp , b) Near jmp , c) far jmp
- d) a and b

مثال في jmp داخل برنامج



- * Obtains a new segment and offset address to accomplish the jump :-

a) Far b) Near c) Short

لأنه ال Far هو الذي ينقل

و مَقْدَسٌ فِي نَفْسِ الرَّحْمَنِ وَ تَجِيءُ فِيهِ IP_{9CS}

* byte 2 and 3 of this 5-byte instruction contain the new offset address

* byte 4 and 5 contain the new seg address

تَرْفِ الدَّيَّانَةَ الْخَلْقَ !

هو الملاحضات التي نكتب فيهم ملاحظات الى د. ج. ما! والله سبحانه وتعالى

صحیح اور خطا اور MCQ

Jumps with Reg operands:-

* Jump can also use a 16 bit Register as an operand

لے 16 bit میں 8 bit ردیالک !!

* An Indirect jump does not add to the In pointer ✓

* JMP BX \rightarrow "indirect" Near \rightarrow 9's

بہجید مائٹسوف ال او کیا ہے Reg

or

mem

Indirect $\frac{1}{100}$

* JMP WORD PTR [BX]

Indirect \Rightarrow memory location \Rightarrow $\bar{v}i$ \Rightarrow $\bar{v}i$.

نویس Near رانہ قال WORD

لَوْ قَالَ: —

JMP DWORD PTR[Bx]

Indirect Location memory location \rightarrow بیان

نویس Far لایف قال DWOR

حزب انگریزوں میں
انقلابی تبدیلی

Ex:- - JMP DX

IF DX = 1234H branches to CS : 1234H,

- JMP WORD PTR [2000H[Bx]]

جولتي

JMP WORD PTR [Bx + 2000H]

if Bx = 1234H

Branches to CS : 5678H

DS: 3234H

DS: 3236H

5678H

- JMP DWORD PTR 2000H[Bx]

Far جولة Far لولسي
CS لولسي

IF Bx = 1234H \Rightarrow Bx + 2000 = 3234H

DS: 3234

5678H

DS: 3236

ABCDH

Branch to location :-

ABCD : 5678

CS \leftarrow \overline{J} \overline{L} IP

code لا high لا

IP لا low لا

!! كنز

- * Direct Jump: The target address is directly given in the In ✓
- * Indirect Jump: The target address is contained in a Reg or memory location ✓

مرات يقولك في الامتحان شن الفرق بين
Far, Near " فهم الـ جواله!!!

تقدر تكتب بالترتيب عاري ولو بتيا تكتب
بال E سوف الـ جواله من الـ الرابع

- 1- Near: الـ CS الـ ديسبلت في نفس الـ CS
 - 1- Far: الـ CS ديسبلت في Seg. جدي
 - 2- Near: The contents of CS is not stored
 - 2- Far: " " " " is also stored along with offset
 - 3- Near: مكنويات الـ SP " stack pointer " تنقص بمقدار 2 وقيمة الـ IP تنقص الـ CS لا
 - 3- Far: مكنويات الـ SP حنقص بمقدار 2 وحنقص الـ CS بعدين تنقص الـ SP مرة ثانية بمقدار 2 وتنقص الـ IP
- اكثر اكرر اكرر!!!

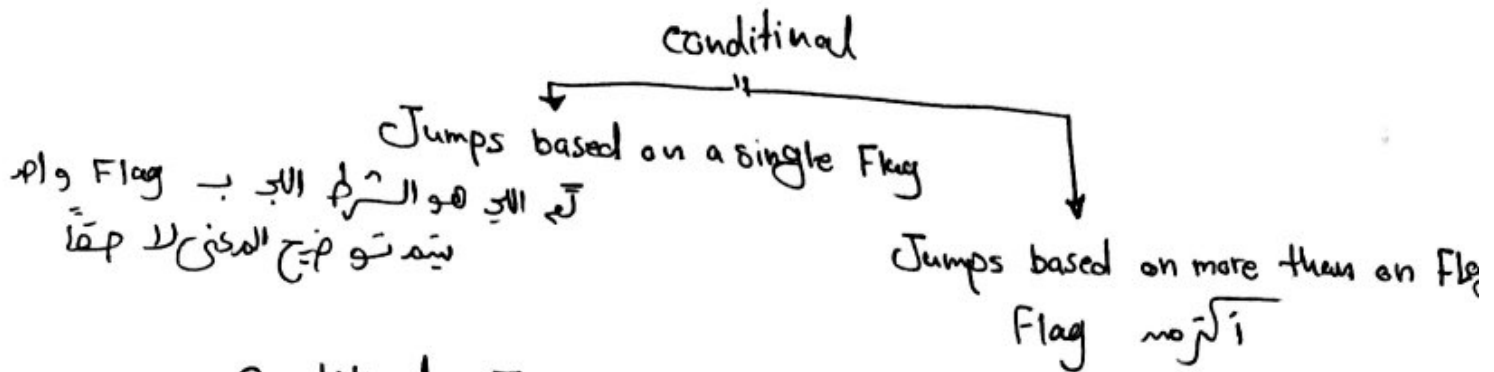
* Three Near Jump and two Far Jump In have the same mnemonic JMP, but they have different opcodes. ✓

opcode " short "	EB	اكثر
opcode " Near "	EG	اكثر
opcode " Far "	EA	اكثر

-: Conditional Jump:-

* Conditional Jump Instruction in 8086 are just 2 bytes long.

ال Jump المشروط الذي منفرضا، لا لو تحقق الشرط 1.



*** Conditional Jump are always short Jump

لهم لأنهم أحسن قننا للشرط في 2 byte فقط
وفي نفس الوقت نفق انه ال Short ال 2 byte

Range « 127 → -128 »

ال Flags الذي حيثشوا في الشرط هما « S, Z, C, O, P »
ال AF مش حيثش لاللا، ر بالك !

* When signed numbers are compared, use the JG, JL

JGE, JLE, JE, JNE
إذا كان أكبر
إذا كان أصغر
إذا كان أكبر
إذا كان أصغر
إذا كان أكبر
إذا كان أصغر

N = NOT

* terms greater than and less than refer to signed

* when unsigned numbers are compared, use the

JA, JB, JAE, JBE, JE, JNE
أعلى
أدنى Below

انتقل Jump based on multiple Flags من السبت الى

الاثنين مهمت 13

سبته انزال سبت ياكوي على
 باقي سبت سبكت و بدم
 ابراج المهمت ، يوم الثلاثاء
 بالتاريخ